6TiSCH Working Group M. Vucinic, Ed.
Internet-Draft Inria
Intended status: Standards Track J. Simon
Expires: May 24, 2019 Analog Devices
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
November 20, 2018

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-09

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, 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.

This Internet-Draft will expire on May 24, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document defines a "secure join" solution for a new device, called "pledge", to securely join a 6TiSCH network. The term "secure join" refers to network access authentication, authorization and parameter distribution, as defined in [I-D.ietf-6tisch-terminology]. The Constrained Join Protocol (CoJP) defined in this document handles parameter distribution needed for a pledge to become a joined node. Authorization mechanisms are considered out of scope. Mutual authentication during network access is achieved through the use of a secure channel, as configured by this document. This document also specifies a configuration of different layers of the 6TiSCH protocol stack that reduces the Denial of Service (DoS) attack surface during the join process.

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 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.

CoJP is a generic protocol that can be used as-is in all modes of IEEE Std 802.15.4, including the Time-Slotted Channel Hopping (TSCH) mode 6TiSCH is based on. CoJP 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 define configuration parameters specific to the technology in question, through companion documents. The overall process described in Section 4 and the configuration of the stack is specific to 6TiSCH.

CoJP assumes the presence of a Join Registrar/Coordinator (JRC), a central entity. The configuration defined in this document assumes that the pledge and the JRC share a secret cryptographic 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 during the provisioning phase or by a key exchange protocol that may precede the execution of CoJP.

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 CoJP messages with the JRC; these messages can be forwarded by nodes already part of the 6TiSCH network, called Join Proxies. The messages exchanged allow the JRC and the pledge to mutually authenticate, based on the properties provided by OSCORE. They also allow the JRC to configure the pledge with link-layer keying material, short identifier 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 [RFC8480] and start sending application traffic.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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)

- o enhanced beacon (EB)

- o join protocol

- o join process

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

- o 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 the network using their 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 generic terms "pledge identifier" and "network identifier". See Section 3.

The terms "secret key" and "symmetric key" are used interchangeably.

3. Provisioning Phase

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:

- pledge identifier. The pledge identifier identifies the (6LBR) pledge. The pledge identifier MUST be unique in the set of all pledge identifiers managed by a JRC. The pledge identifier uniqueness is an important security requirement, as discussed in Section 9. The pledge identifier is typically the globally unique 64-bit Extended Unique Identifier (EUI-64) of the IEEE Std 802.15.4 device, in which case it is provisioned by the hardware manufacturer. The pledge 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 (see Section 10), it is possible to use a pledge identifier different from the EUI-64. For example, a pledge identifier may be a random byte string, but care needs to be taken that such a string meets the uniqueness requirement.

- Pre-Shared Key (PSK). A secret cryptographic key shared between the (6LBR) pledge and the JRC. The JRC additionally needs to store the pledge identifier bound to the given PSK. Each (6LBR) pledge MUST be provisioned with a unique PSK. The PSK SHOULD be a cryptographically strong key, at least 128 bits in length, indistinguishable by feasible computation from a random uniform string of the same length. How the PSK is generated and/or provisioned is out of scope of this specification. This could be done during a provisioning step or companion documents can specify the use of a key agreement protocol. Common pitfalls when generating PSKs are discussed in Section 9.

- Optionally, a network identifier. The network identifier identifies the 6TiSCH network. The network identifier MUST be carried within Enhanced Beacon (EB) frames. Typically, the 16-bit Personal Area Network Identifier (PAN ID) defined in [IEEE802.15.4] is used as the network identifier. However, PAN ID is not considered a stable network identifier as it may change during network lifetime if a collision with another network is detected. Companion documents can specify the use of a different network identifier for join purposes, but this is out of scope of this specification. Provisioning the network identifier is RECOMMENDED. However, due to 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 advertised network at a time, which significantly prolongs the join process. This parameter MUST be provisioned to the 6LBR pledge.

- Optionally, any non-default algorithms. The default algorithms are specified in Section 7.3.3. 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 through CoJP.

4. 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 acts as a 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.

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

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

4.1. 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, the value of the Join Metric field within EBs, 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 8.1.1.

Once the pledge selects the EB, it synchronizes to it and transitions into a low-power mode. It follows the schedule information contained in the EB 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 acts as the JP.

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

4.2. 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 5.

4.3. 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 end-to-end channel that provides confidentiality, data authenticity and replay protection. Frames carrying CoJP messages are not protected with link-layer security when exchanged between the pledge and the JP as the pledge is not in possession of the link-layer keys in use. How JP and pledge accept these unprotected frames is discussed in Section 5. When frames carrying CoJP messages are exchanged between nodes that have already joined the network, the link-layer security is applied according to the security configuration used in the network.

4.3.1. 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 links. How exactly this happens is out of scope of this document; some networks may wish to dedicate specific link layer resources for this join traffic.

4.3.2. 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 the JP using the operating routes in the network. The JP delivers it to the pledge. The JP operates as the application-layer proxy.

The Join Response contains different parameters needed by the pledge to become a fully operational network node. These parameters include the link-layer key(s) currently in use in the network, the short address assigned to the pledge, the IPv6 address of the JRC needed by the pledge to operate as the JP, among others.

4.4. The Special Case of the 6LBR Pledge Joining

The 6LBR pledge performs Section 4.3 of the join process described above, just as any other pledge, albeit over a different network interface. There is no JP intermediating the communication between the 6LBR pledge and the JRC, as described in Section 6. 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.

5. 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 DoS vector, as discussed in Section 9.

6. 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.

Once the pledge obtains link-layer keys and becomes a joined node, it is able to securely communicate with its neighbors, obtain the network IPv6 prefix and form its global IPv6 address. The joined node then undergoes an independent process to bootstrap its neighbor cache entries, possibly with a node that formerly acted as a JP, following [RFC6775]. From the point of view of the JP, there is no relationship between the neighbor cache entry belonging to a pledge and the joined node that formerly acted as a pledge.

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 7.

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 IPv6 address of the JRC from the Join Response, as specified in Section 8.1.2; 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.

6.1. Identification of Unauthenticated Traffic

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

When operating as part of a [RFC8180] 6TiSCH minimal network using distributed scheduling algorithms, the traffic from unauthenticated pledges 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 originates from unauthenticated pledges, and so can avoid allocating additional bandwidth itself. The Join Proxy implements a data cap on outgoing join traffic through CoAP's congestion control mechanism. This cap will not protect intermediate nodes as they can not tell join traffic from regular traffic. Despite the data cap implemented separately on each Join Proxy, the aggregate join 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 traffic originated at unauthenticated pledges. 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). Based on the DSCP, intermediate nodes can decide whether to act on a given packet.

6.1.1. Traffic from JP to JRC

The Join Proxy SHOULD set the DSCP of packets that it produces as part of the forwarding 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.

6.1.2. Traffic from JRC to JP

The JRC SHOULD set the DSCP of join response packets addressed to the Join Proxy to AF42 code point. AF42 has lower drop probability than AF43, giving this traffic priority in buffers over the traffic going towards the JRC.

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.

7. 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 acts as a CoAP client; the JRC acts as 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.

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 8.1.1.

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.

7.1. Statelessness of the JP

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 client, and the UDP source port number. Since the JP can be a constrained device that acts as a CoAP proxy, memory limitations make it prone to a Denial-of-Service (DoS) attack.

This DoS vector on the JP can be mitigated by making the JP act as a stateless CoAP proxy, where "state" refers to individual pledges. The JP can wrap the state it needs to keep for a given pledge throughout the network stack in a "state object" and include it as a CoAP token in the forwarded request to the JRC. The JP may use the CoAP token as defined in [RFC7252], if the size of the serialized state object permits, or use the extended CoAP token defined in [I-D.hartke-core-stateless], to transport the state object. Since

the CoAP token is echoed back in the response, the JP is able to decode the state object and configure the state needed to forward the response to the pledge. The information that the JP needs to encode in the state object to operate in a fully stateless manner with respect to a given pledge is implementation specific.

It is RECOMMENDED that the JP operates in a stateless manner and signals the per-pledge state within the CoAP token, for every request it forwards into the network on behalf of unauthenticated pledges. When operating in a stateless manner, the state object communicated in the token MUST be integrity protected, potentially with a key that is known only to the JP, MUST include a freshness indicator, and MAY be encrypted. Security considerations from [I-D.hartke-core-stateless] apply.

When operating in a stateless manner, the type of the CoAP message that the JP forwards on behalf of the pledge MUST be non-confirmable (NON), regardless of the message type received from the pledge. The use of a non-confirmable message by the JP alleviates the JP from keeping CoAP message exchange state. The retransmission burden is then entirely shifted to the pledge. A JP that operates in a stateless manner still needs to keep congestion control state with the JRC, see Section 9. Recommended values of CoAP settings for use during the join process, both by the pledge and the JP, are given in Section 7.2.

Note that in some networking stack implementations, a fully (per-pledge) stateless operation of the JP may be challenging from the implementation's point of view. In those cases, the JP may operate as a statefull proxy that stores the per-pledge state until the response is received or timed out, but this comes at a price of an additional DoS vector.

7.2. Recommended Settings

This section gives RECOMMENDED values of CoAP settings during the join process.

Name	Default Value: Pledge	Default Value: JP
ACK_TIMEOUT	10 seconds	(10 seconds)
ACK_RANDOM_FACTOR	1.5	(1.5)
MAX_RETRANSMIT	4	(4)

Recommended CoAP settings. Values enclosed in () have no effect when JP operates in a stateless manner.

These values 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.

The PROBING_RATE value at the JP is controlled by the join rate parameter, see Section 8.4.2. Following [RFC7252], the average data rate in sending to the JRC must not exceed PROBING_RATE. For security reasons, the average data rate SHOULD be measured over a rather short window, e.g. ACK_TIMEOUT, see Section 9.

7.3. OSCORE

Before the (6LBR) pledge and the JRC start exchanging CoAP messages protected with OSCORE, they need to derive the OSCORE security context from the provisioned parameters, as discussed in Section 3.

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 the empty byte string.

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

- the ID of the pledge MUST be set to the empty byte string. This identifier is used as the OSCORE Sender ID of the pledge in the security context derivation, since the pledge initially acts as 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 of the pledge in the security context derivation, as the JRC initially acts as a CoAP server.

o 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.

o the Key Derivation Function MUST be agreed out-of-band by the same mechanism used to provision the PSK. Default is HKDF SHA-256 [RFC5869].

Since the pledge's OSCORE ID is the empty byte string, when constructing the OSCORE option, the pledge sets the k bit in the OSCORE flag byte, but indicates a 0-length kid. The pledge transports its pledge identifier within the kid context field of the OSCORE option. The derivation in [I-D.ietf-core-object-security] results in OSCORE 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. For details on nonce and OSCORE option construction, refer to [I-D.ietf-core-object-security].

Implementations MUST ensure that multiple CoAP requests to different JRCs are properly incrementing the sequence numbers in the OSCORE security context for each message, so that the same sequence number is never reused in distinct requests. 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 security guarantees of the Authenticated Encryption with Associated Data (AEAD) algorithm because of 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 8.2. The (6LBR) pledge and the JRC use the OSCORE security context parameters (e.g. sender and recipient identifiers) as they were used at the moment of context derivation, regardless of whether they currently act as a CoAP client or a CoAP server. A (6LBR) pledge is expected to have exactly one OSCORE security context with the JRC.

7.3.1. Replay Window and Persistency

Both (6LBR) pledge and the JRC MUST implement a replay protection mechanism. The use of the default OSCORE replay protection mechanism specified in Section 3.2.2 of [I-D.ietf-core-object-security] is RECOMMENDED.

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 7.5.1 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.

7.3.2. OSCORE Error Handling

Errors raised by OSCORE during the join process MUST be silently dropped, with no error response being signaled. The pledge MUST silently discard any response not protected with OSCORE, including error codes.

Such errors 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 OSCORE messages prevents a DoS attack on the pledge where the attacker could send bogus error responses, forcing the pledge to attempt joining one network at a time, until all networks have been tried.

7.3.3. 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. See Appendix B for implementation guidance when code footprint is important.

8. Constrained Join Protocol (CoJP)

The 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         |  |
+-----------------------------------+  /
+-----------------------------------+
|                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:

o 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 8.1.1.

o 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 8.1.2.

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:

- o 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 8.2.1.

- o 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 8.2.2.

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 8.4.

8.1. Join Exchange

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

8.1.1. Join Request Message

The Join Request message that the (6LBR) pledge sends SHALL be mapped to a CoAP request:

- o The request method is POST.

- o The type is Confirmable (CON).

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

- o 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.

- o The Uri-Path option is set to "j".

- o The OSCORE option SHALL be set according to [I-D.ietf-core-object-security]. The OSCORE security context used is the one derived in Section 7.3. The OSCORE kid context allows the JRC to retrieve the security context for a given pledge.

- o The payload is a Join_Request CBOR object, as defined in Section 8.4.1.

Since the Join Request is a confirmable message, the transmission at (6LBR) pledge will be controlled by CoAP's retransmission mechanism. The JP, when operating in a stateless manner, forwards this Join Request as a non-confirmable (NON) CoAP message, as specified in Section 7. If the CoAP at (6LBR) pledge declares the message transmission as failure, the (6LBR) pledge SHOULD attempt to join the next advertised 6TiSCH network. See Section 7.2 for recommended values of CoAP settings to use during the join exchange.

If all join attempts to advertised networks have failed, the (6LBR) pledge SHOULD signal to the user the presence of an error condition, through some out-of-band mechanism.

8.1.2. Join Response Message

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

- o The response Code is 2.04 (Changed).

- o The payload is a Configuration CBOR object, as defined in Section 8.4.2.

8.2. 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 becomes a CoAP 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.

8.2.1. 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).

- o The Uri-Path option is set to "j".

- o The OSCORE option SHALL be set according to [I-D.ietf-core-object-security]. The OSCORE security context used is the one derived in Section 7.3. 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.

- o The payload is a Configuration CBOR object, as defined in Section 8.4.2.

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.

In case the JRC does not receive a response to a Parameter Update message, it attempts multiple retransmissions, as configured by the underlying CoAP retransmission mechanism triggered for confirmable messages. Finally, if the CoAP implementation declares the transmission as failure, the JRC may consider this as a hint that the joined node is no longer in the network. How the JRC decides when to stop attempting to contact a previously joined node is out of scope of this specification but security considerations on the reuse of assigned resources apply, as discussed in Section 9.

8.2.2. Parameter Update Response Message

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

- o The response Code is 2.04 (Changed).

- o The payload is empty.

8.3. Error Handling

8.3.1. CoJP CBOR Object Processing

This section describes error handling when processing CoJP CBOR objects that are transported within the payload of different CoJP messages. See Section 7.3.2 for the handling of errors that may be raised by the underlying OSCORE implementation.

CoJP CBOR objects are transported within both CoAP requests and responses. When an error is detected while processing CoJP objects in a CoAP request (Join Request message, Parameter Update message), an Error Response message MUST be returned. An Error Response message maps to a CoAP response and is specified in Section 8.3.2.

When an error is detected while processing a CoJP object in a CoAP response (Join Response message), a (6LBR) pledge SHOULD reattempt to join. In this case, the (6LBR) pledge SHOULD include the Error CBOR object within the Join Request object in the following Join Request message. A (6LBR) pledge MUST NOT attempt more than MAX_RETRANSMIT number of attempts to join if the processing of the Join Response message fails each time. If COJP_MAX_JOIN_ATTEMPTS number of attempts is reached without success, the (6LBR) pledge SHOULD signal to the user the presence of an error condition, through some out-of-band mechanism.

8.3.2. Error Response Message

The Error Response Message is returned for any CoJP request when the processing of the payload failed. The Error Response message is protected by OSCORE as any other CoJP protocol message.

The Error Response message SHALL be mapped to a CoAP response:

- o The response Code is 4.00 (Bad Request).

- o The payload is an Error CBOR object, as defined in Section 8.4.5, containing the error code that triggered the sending of this message.

8.3.3. Failure Handling

The Parameter Update exchange may be triggered at any time during the network lifetime, which may span several years. During this period, it may occur that a joined node or the JRC experience unexpected events such as reboots or complete failures.

This document mandates that the mutable parameters in the security context are written to persistent memory (see Section 7.3.1) by both the JRC and pledges (joined nodes). In case of a reboot on either side, the retrieval of mutable security context parameters is feasible from the persistent memory such that there is no risk of AEAD nonce reuse due to a reinitialized Sender Sequence number, or of a replay attack due to the reinitialized replay window.

In the case of a complete failure, where the mutable security context parameters cannot be retrieved, it is expected that a failed joined node is replaced with a new physical device, using a new pledge identifier and a PSK. When such an event occurs at the JRC, it is likely that the information about joined nodes, their assigned short identifiers and mutable security context parameters is lost. If this is the case, during the process of JRC replacement, the network administrator MUST force all the networks managed by the failed JRC to rejoin, through e.g. the reinitialization of the 6LBR nodes. Since the joined nodes kept track of their mutable security context parameters, they will use these during the (re)join exchange without a risk of AEAD nonce reuse. However, even after all the nodes rejoined, the AEAD nonce reuse risk exists during the first Parameter Update exchange, as the new JRC does not possess the last Sender Sequence number used, and can only initialize it to zero. Since the sending of this first Parameter Update message by the new JRC results in AEAD nonce reuse, the JRC MUST set the payload to a randomly generated byte string, at least 40 bytes long.

When such a message arrives at the joined node, the OSCORE implementation rejects it due to the Partial IV being largely below the acceptable replay window state and does not process the payload. When this is detected, the joined node MUST send a 4.01 Unauthorized response, as per Section 7.4 of [I-D.ietf-core-object-security]. The payload of the response MUST be the Error object specified in Section 8.4.5, with error code set to "Significant OSCORE partial IV mismatch" from Table 4 and Additional information set to the next Partial IV the joined node will expect. When protecting this error response by OSCORE, the joined node MUST use its Sender Sequence number to generate a new nonce and include the corresponding Partial IV in the CoAP OSCORE option, as detailed in Section 8.3 of [I-D.ietf-core-object-security]. Upon successful OSCORE verification of the received CoJP message, the JRC processes the error response and configures the Sender Sequence number to the one indicated in the Additional information field. The next Parameter Update exchange triggered by the JRC will therefore use the proper Sender Sequence number and will be accepted by the joined node.

8.4. 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.

8.4.1. 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 labels can be found in the "CoJP Parameters" registry Section 11.1.

- 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. 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 regardless of the role parameter value.

- response processing error: The identifier of the error from the previous join attempt, encoded as an Error object described in Section 8.4.5. This parameter MAY be included. If a (6LBR) pledge previously attempted to join and received a valid Join Response message over OSCORE, but failed to process its payload (Configuration object), it SHOULD include this parameter to facilitate the debugging process.

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

```
Join_Request = {
    ? 1 : uint,              ; role
    ? 5 : bstr,              ; network identifier
    ? 8 : Error,             ; response processing error
}
```

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.

8.4.2. 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 labels can be found in "CoJP Parameters" registry Section 11.1.

- 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 8.4.3. 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 immediately install and start using the new keys for all outgoing traffic, and remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed. 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. 6LBR and non-6LBR nodes accept any frame for which they have 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 start using the keys from the new set for all outgoing traffic. A non-6LBR node MUST remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed. 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 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.

- short identifier: a compact identifier assigned to the pledge. The short identifier structure is described in Section 8.4.4. The short identifier parameter MAY be included in a Configuration object.

- 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 from 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 Configuration object, 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 6.

- blacklist: An array encompassing a list of pledge identifiers that are blacklisted by the JRC, with each pledge identifier encoded as a byte string. The blacklist parameter MAY be included in a Configuration object. When present, the array MUST contain zero or more byte strings encoding pledge identifiers. The joined node MUST silently drop any link-layer frames originating from the pledge identifiers enclosed in the blacklist parameter. When this parameter is received, its value MUST overwrite any previously set values. This parameter allows the JRC to configure the node acting as a JP to filter out traffic from misconfigured or malicious pledges before their traffic is forwarded into the network. If the JRC decides to remove a given pledge identifier from a blacklist, it omits the pledge identifier in the blacklist parameter value it sends next.

- join rate: Average data rate of join traffic forwarded into the network that should not be exceeded when a joined node operates as

a JP, encoded as an unsigned integer in bytes per second. The join rate parameter MAY be included in a Configuration object. This parameter allows the JRC to configure different nodes in the network to operate as JP, and act in case of an attack by throttling the rate at which JP forwards unauthenticated traffic into the network. When this parameter is present in a Configuration object, the value MUST be used to set the PROBING_RATE of CoAP at the joined node for communication with the JRC. In case this parameter is set to zero, a joined node MUST silently drop any join traffic coming from unauthenticated pledges. In case this parameter is omitted, the value of positive infinity SHOULD be assumed. Node operating as a JP MAY use another mechanism that is out of scope of this specification to configure PROBING_RATE of CoAP in the absence of join rate parameter from the Configuration object.

The CDDL fragment that represents the text above for the Configuration follows. Structures Link_Layer_Key and Short_Identifier are specified in Section 8.4.3 and Section 8.4.4.

```
Configuration = {
    ? 2 : [ +Link_Layer_Key ],   ; link-layer key set
    ? 3 : Short_Identifier,      ; short identifier
    ? 4 : bstr,                  ; JRC address
    ? 6 : [ *bstr ],             ; blacklist
    ? 7 : uint                   ; join rate
}
```

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]]
short identifier	3	array	Identifies the assigned short identifier	[[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]]
blacklist	6	array	Identifies the blacklist parameter	[[this document]]
join rate	7	unsigned integer	Identifier the join rate parameter	[[this document]]
error	8	array	Identifies the error parameter	[[this document]]

Table 2: CoJP parameters map labels.

8.4.3. Link-Layer Key

The Link_Layer_Key structure encompasses the parameters needed to configure the link-layer security module: the key identifier; the value of the cryptographic key; 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; and any additional information that may be needed to configure the key.

For encoding compactness, the Link_Layer_Key object is not enclosed in a top-level CBOR object. Rather, it is transported as a sequence of CBOR elements, some being optional.

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

o key_id: The identifier of the key, encoded as a CBOR unsigned integer. This parameter MUST be included. If the decoded CBOR unsigned integer value is larger than the maximum link-layer key identifier, the key is considered invalid. In case the key is considered invalid, the key MUST be discarded and the implementation MUST signal the error as specified in Section 8.3.1.

o 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 an integer. This parameter MAY be included. Possible values and the corresponding link-layer settings are specified in IANA "CoJP Key Usage" registry (Section 11.2). In case the parameter is omitted, the default value of 0 from Table 3 MUST be assumed.

o 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 implementation MUST signal the error as specified in Section 8.3.1.

o key_addinfo: Additional information needed to configure the link-layer key, encoded as a byte string. This parameter MAY be included. The processing of this parameter is dependent on the link-layer technology in use and a particular keying mode.

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, a uint that follows a byte string signals to the decoder that a new Link_Layer_Key object is being processed.

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

```
Link_Layer_Key = (
      key_id             : uint,
    ? key_usage          : int,
      key_value          : bstr,
    ? key_addinfo        : bstr,
)
```

Name	Value	Algorithm	Description	Reference
6TiSCH-K1K2-ENC-MIC32	0	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC64	1	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC128	2	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC32	3	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC64	4	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC128	5	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1-MIC32	6	IEEE802154-AES-CCM-128	Use MIC-32 for EBs.	[[this document]]
6TiSCH-K1-MIC64	7	IEEE802154-AES-	Use MIC-64	[[this d

		CCM-128	for EBs.	ocument]]
6TiSCH-K1-MIC128	8	IEEE802154-AES-CCM-128	Use MIC-128 for EBs.	[[this document]]
6TiSCH-K2-MIC32	9	IEEE802154-AES-CCM-128	Use MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-MIC64	10	IEEE802154-AES-CCM-128	Use MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-MIC128	11	IEEE802154-AES-CCM-128	Use MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC32	12	IEEE802154-AES-CCM-128	Use ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC64	13	IEEE802154-AES-CCM-128	Use ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC128	14	IEEE802154-AES-CCM-128	Use ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]

Table 3: Key Usage values.

8.4.3.1. Use in IEEE Std 802.15.4

When Link_Layer_Key is used in the context of [IEEE802.15.4], the following considerations apply.

Signaling of different keying modes of [IEEE802.15.4] is done based on the parameter values present in a Link_Layer_Key object.

- o Key ID Mode 0x00 (Implicit, pairwise): key_id parameter MUST be set to 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to the link-layer address(es) of a single peer with whom the key should be used. Depending on the configuration of the network, key_addinfo may carry the peer's long link-layer address (i.e. pledge identifier), short link-layer address, or their concatenation with the long address being encoded first. Which address is carried is determined from the length of the byte string.

- o Key ID Mode 0x01 (Key Index): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST NOT be present.

- o Key ID Mode 0x02 (4-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 4 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

- o Key ID Mode 0x03 (8-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 8 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

In all cases, key_usage parameter determines how a particular key should be used in respect to incoming and outgoing security policies.

For Key ID Modes 0x01 - 0x03, parameter key_id sets the "secKeyIndex" parameter of {{IEEE802.15.4} that is signaled in all outgoing frames secured with a given key. The maximum value key_id can have is 254. The value of 255 is reserved in {{IEEE802.15.4} and is therefore considered invalid.

Key ID Mode 0x00 (Implicit, pairwise) enables the JRC to act as a trusted third party and assign pairwise keys between nodes in the network. How JRC learns about the network topology is out of scope of this specification, but could be done through 6LBR - JRC signaling for example. Pairwise keys could also be derived through a key agreement protocol executed between the peers directly, where the authentication is based on the symmetric cryptographic material provided to both peers by the JRC. Such a protocol is out of scope of this specification.

8.4.4. Short Identifier

The Short_Identifier object represents an identifier assigned to the pledge. It is encoded as a CBOR array object, containing, in order:

- identifier: The short identifier assigned to the pledge, encoded as a byte string. This parameter MUST be included. The identifier MUST be unique in the set of all identifiers assigned in a network that is managed by a JRC. In case the identifier is invalid, the decoder MUST silently ignore the Short_Identifier object.

- lease_time: The validity of the identifier in hours 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 identifier 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_Identifier in the Configuration object, as described in Section 8.2. In case the lease expires, the node SHOULD initiate a new join exchange, as described in Section 8.1. In case this parameter is omitted, the value of positive infinity MUST be assumed, meaning that the identifier is valid for as long as the node participates in the network.

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

```
Short_Identifier = [
      identifier        : bstr,
    ? lease_time        : uint
]
```

8.4.4.1. Use in IEEE Std 802.15.4

When Short_Identifier is used in the context of [IEEE802.15.4], the following considerations apply.

The identifier MUST be used to set the short address of IEEE Std 802.15.4 module. When operating in TSCH mode, the identifier MUST be unique in the set of all identifiers assigned in multiple networks that share link-layer key(s). If the length of the byte string corresponding to the identifier parameter is different than 2, the identifier is considered invalid. The values 0xfffe and 0xffff are reserved by [IEEE802.15.4] and their use is considered invalid.

The security properties offered by the [IEEE802.15.4] link-layer in TSCH mode are conditioned on the uniqueness requirement of the short identifier (i.e. short address). The short address is one of the inputs in the construction of the nonce, which is used to protect link-layer frames. If a misconfiguration occurs, and the same short address is assigned twice under the same link-layer key, the loss of security properties is eminent. For this reason, practices where the pledge generates the short identifier locally are not safe and are likely to result in the loss of link-layer security properties.

The JRC MUST ensure that at any given time there are never two same short identifiers being used under the same link-layer key. If the lease_time parameter of a given Short_Identifier object is set to positive infinity, care needs to be taken that the corresponding identifier is not assigned to another node until the JRC is certain that it is no longer in use, potentially through out-of-band signaling. If the lease_time parameter expires for any reason, the JRC should take into consideration potential ongoing transmissions by the joined node, which may be hanging in the queues, before assigning the same identifier to another node.

8.4.5. Error Object

The Error object is encoded as a CBOR array object, containing in order:

o error_code: Error code for the first encountered error while processing a CoJP object, encoded as an integer. This parameter MUST be included. Possible values of this parameter are specified in the IANA "CoJP Error Registry" (Section 11.3).

o error_addinfo: Additional information relevant to the error. This parameter MUST be included. This parameter MUST be set as described by the "Additional info" column of the "CoJP Error Registry" (Section 11.3).

o error_description: Human-readable description of the error, encoded as a text string. This parameter MAY be included. The RECOMMENDED setting of this parameter is the "Description" column of the "CoJP Error Registry" Section 11.3).

The CDDL fragment that represents the text above for the Error object follows.

```
Error = [
      error_code        : int,
      error_addinfo     : int / bstr / tstr / nil,
    ? error_description : tstr,
]
```

Description	Value	Additional info	Additional info type	Reference
Invalid Join_Request object	0	None	nil	[[this document]]
Invalid Configuration object	1	None	nil	[[this document]]
Invalid parameter	2	Label of the invalid parameter	int	[[this document]]
Invalid link-layer key	3	Index of the invalid key	uint	[[this document]]
Significant OSCORE partial IV mismatch	4	Next acceptable OSCORE partial IV	bstr	[[this document]]

Table 4: CoJP error codes.

8.5. Recommended Settings

This section gives RECOMMENDED values of CoJP settings discussed in this section.

Name	Default Value
COJP_MAX_JOIN_ATTEMPTS	4
COJP_REKEYING_GUARD_TIME	12 seconds

Recommended CoJP settings.

The COJP_REKEYING_GUARD_TIME value SHOULD take into account possible retransmissions at the link layer due to imperfect wireless links.

9. Security Considerations

Since this document uses the pledge identifier to set the ID Context parameter of OSCORE, an important security requirement is that the pledge identifier is unique in the set of all pledge identifiers managed by a JRC. The uniqueness of the pledge identifier ensures unique (key, nonce) pairs for AEAD algorithm used by OSCORE. It also allows the JRC to retrieve the correct security context, upon the reception of a Join Request message. The management of pledge identifiers is simplified if the globally unique EUI-64 is used, but this comes with privacy risks, as discussed in Section 10.

This document further mandates that the (6LBR) pledge and the JRC are provisioned with unique PSKs. The PSK is used to set the OSCORE Master Secret during security context derivation. This derivation process results in OSCORE keys that are 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.

Many vendors are known to use unsafe practices when generating and provisioning PSKs. The use of a single PSK shared among a group of devices is a common pitfall that results in poor security. In this case, the compromise of a single device is likely to lead to a compromise of the entire batch, with the attacker having the ability to impersonate a legitimate device and join the network, generate bogus data and disturb the network operation. As a reminder, recall the well-known problem with Bluetooth headsets with a "0000" pin. Additionally, some vendors use methods such as scrambling or hashing of device serial numbers or their EUI-64 to generate "unique" PSKs. Without any secret information involved, the effort that the attacker needs to invest into breaking these unsafe derivation methods is quite low, resulting in the possible impersonation of any device from the batch, without even needing to compromise a single device. The use of cryptographically secure random number generators to generate the PSK is RECOMMENDED, see [NIST800-90A] for different mechanisms using deterministic methods.

The JP forwards the unauthenticated join traffic into the network. A data cap on the JP prevents it from forwarding more traffic than the network can handle. The data cap can be configured by the JRC by including a join rate parameter in the Join Response and it is implemented through the CoAP's PROBING_RATE setting. The use of a data cap at a JP forces attackers to use more than one JP 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 a DoS attack 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. Note that this temporary blacklist is different from the one communicated as part of the CoJP Configuration object as it helps pledge fight a DoS attack. These bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode. The blacklist communicated as part of the CoJP Configuration object helps JP fight a DoS attack by a malicious pledge.

10. Privacy Considerations

The join solution specified in this document relies on the uniqueness of the pledge identifier in the set of all pledge identifiers managed by a 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 process occurs 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 reduces 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. However, an eavesdropper with access to the radio medium

during the join process may be able to correlate the assigned short address with the extended address based on timing information with a non-negligible probability. This probability decreases with an increasing number of pledges joining concurrently.

11. 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.

11.1. 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 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.

11.2. 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.

Reference: 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.

11.3. CoJP Error 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 Error Registry".

The columns of this registry are:

Description: This is a descriptive human-readable name. The description MUST be unique. It is not used in the encoding.

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

Additional information: This is a descriptive name of additional information that is meaningful for the error. The name is not used in the encoding.

Additional information type: A CBOR type of the additional information field.

Reference: 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 4.

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.

12. 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 agreements: No 644852, project ARMOUR; No 687884, project F-Interop and open-call project SPOTS; No 732638, project Fed4FIRE+ and open-call project SODA.

The following individuals provided input to this document (in alphabetic order): Tengfei Chang, Klaus Hartke, Tero Kivinen, Jim Schaad, Goeran Selander, Yasuyuki Tanaka, Pascal Thubert, William Vignat, Xavier Vilajosana, Thomas Watteyne.

13. References

13.1. 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-15 (work in progress), August 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>.

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

13.2. Informative References

[I-D.hartke-core-stateless] Hartke, K., "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", draft-hartke-core-stateless-02 (work in progress), October 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-15 (work in progress), October 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 and JSON data structures", draft-ietf-cbor-cddl-06 (work in progress), November 2018.

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

[NIST800-90A]
NIST Special Publication 800-90A, Revision 1, ., Barker, E., and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", 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>.

[RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <https://www.rfc-editor.org/info/rfc8480>.

Appendix A. Example

Figure 3 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the OSCORE 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 sets the CoAP token 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. The 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 state within the CoAP token before deciding where to forward. The JP sets its internal state to that found in the token, 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: -
      +--------->|          |    Proxy-Scheme: coap
      |          |          |        Uri-Host: 6tisch.arpa
      |          |          |          OSCORE: kid: -,
      |          |          |                  kid_context: EUI-64,
      |          |          |                  Partial IV: 1
      |          |          |         Payload: { Code: 0.02 (POST),
      |          |          |                    Uri-Path: "j",
      |          |          |                    join_request, <Tag> }
      |          |          |
      |          |  Join    |            Code: 0.02 (POST)
      |          | Request  |           Token: opaque state
      |          +--------->|          OSCORE: kid: -,
      |          |          |                  kid_context: EUI-64,
      |          |          |                  Partial IV: 1
      |          |          |         Payload: { Code: 0.02 (POST),
      |          |          |                    Uri-Path: "j",
      |          |          |                    join_request, <Tag> }
      |          |          |
      |          |          |
      |          |  Join    |            Code: 2.04 (Changed)
      |          | Response |           Token: opaque state
      |          |<---------+          OSCORE: -
      |          |          |         Payload: { Code: 2.04 (Changed),
      |          |          |                    configuration, <Tag> }
      |          |          |
      |          |          |
      |  Join    |          |            Code: 2.04 (Changed)
      | Response |          |           Token: -
      |<---------+          |          OSCORE: -
      |          |          |         Payload: { Code: 2.04 (Changed),
      |          |          |                    configuration, <Tag> }
      |          |          |
```

Figure 3: Example of a successful join protocol exchange. { ... } denotes authenticated encryption, <Tag> denotes the authentication tag.

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_id /
          h'e6bf4287c2d7618d6a9687445ffd33e6' / key_value /
        ],
    3 : [           / short identifier /
          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-MIC32" is implied. Since key_addinfo parameter is not present and key_id is different than 0, Key ID Mode 0x01 (Key Index) is implied. Similarly, since the lease_time parameter is not present in the short identifier object, the default value of positive infinity is implied.

The configuration object encodes to

h'a202820150e6bf4287c2d7618d6a9687445ffd33e6038142af93' with a size of 26 bytes.

Appendix B. Lightweight Implementation Option

In environments where optimizing the implementation footprint is important, it is possible to implement this specification without having the implementations of HKDF [RFC5869] and SHA [RFC4231] on constrained devices. HKDF and SHA are used during the OSCORE security context derivation phase. This derivation can also be done by the JRC or a provisioning device, on behalf of the (6LBR) pledge during the provisioning phase. In that case, the derived OSCORE security context parameters are written directly into the (6LBR) pledge, without requiring the PSK be provisioned to the (6LBR) pledge.

The use of HKDF to derive OSCORE security context parameters ensures that the resulting OSCORE keys have good security properties, and are unique as long as the input for different pledges varies. This specification ensures the uniqueness by mandating unique pledge identifiers and a unique PSK for each (6LBR) pledge. From the AEAD nonce reuse viewpoint, having a unique pledge identifier is a sufficient condition. However, as discussed in Section 9, the use of a single PSK shared among many devices is a common security pitfall. The compromise of this shared PSK on a single device would lead to the compromise of the entire batch. When using the implementation/ deployment scheme outlined above, the PSK does not need to be written to individual pledges. As a consequence, even if a shared PSK is used, the scheme offers the same level of security as in the scenario where each pledge is provisioned with a unique PSK.

Authors' Addresses

Malisa Vucinic (editor)
Inria
2 Rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

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