RFC9031: Constrained Join Protocol (CoJP) for 6TiSCH

Mališa Vučinić, Jonathan Simon, Kristofer S.J. Pister, Michael Richardson

Stream: Internet Engineering Task Force (IETF)
RFC: 9031
Category: Standards Track
Published: May 2021
ISSN: 2070-1721
Authors:
M. Vučinić, Ed. *Inria*
J. Simon *Analog Devices*
K. Pister *University of California Berkeley*
M. Richardson *Sandelman Software Works*

RFC 9031
Constrained Join Protocol (CoJP) for 6TiSCH

Abstract

This document describes the minimal framework required for a new device, called a "pledge", to securely join a 6TiSCH (IPv6 over the Time-Slotted Channel Hopping mode of IEEE 802.15.4) 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 it describes how to configure 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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9031.

Copyright Notice

Table of Contents

1. Introduction

This document defines a "secure join" solution for a new device, called a "pledge", to securely join a 6TiSCH network. The term "secure join" refers to network access authentication, authorization, and parameter distribution as defined in [RFC9030]. The Constrained Join Protocol (CoJP) defined in this document handles parameter distribution needed for a pledge to become a joined node. Mutual authentication during network access and implicit authorization are achieved through the use of a secure channel as configured according to 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 in order 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 microcontrollers at the heart of 6TiSCH nodes have small amounts 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 [RFC8613] 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 [IEEE802.15.4], including the Time-Slotted Channel Hopping (TSCH) mode on which 6TiSCH is based. CoJP may also 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 through companion documents the configuration parameters specific to the technology in question. The overall process is 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 unique symmetric 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; for this end-to-end communication to happen, the messages are forwarded by nodes, called Join Proxies, that are already part of the 6TiSCH network. 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, a 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 6TiSCH Operation Sublayer (6top) Protocol [RFC8480] and can 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 BCP 14 [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 [RFC9030], [RFC7252], [RFC8613], and [RFC8152].

The specification also includes a set of informative specifications using the Concise Data Definition Language (CDDL) [RFC8610].

The following terms defined in [RFC9030] 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 [RFC8505] are also used throughout this document:

- 6LoWPAN Border Router (6LBR)
- 6LoWPAN Node (6LN)

The term "6LBR" is used interchangeably with the term "DODAG root" defined in [RFC6550] on the assumption that the two entities are co-located, as recommended by [RFC9030].

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 applies equally 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.

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 with a number of mechanisms, such as using a JTAG (Joint Test Action Group) interface, using a serial (craft) console interface, pushing buttons simultaneously on different devices, configuring over-the-air 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 are 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. Depending on the configuration, the pledge identifier may also be used after the join process to identify the joined node. 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 symmetric cryptographic key shared between the (6LBR) pledge and the JRC. To look up the PSK for a given pledge, the JRC additionally needs to store the corresponding pledge identifier. Each (6LBR) pledge **MUST** be provisioned with a unique PSK.

The PSK **MUST** be a cryptographically strong key, with at least 128 bits of entropy, 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. In the case of recommissioning a device to a new owner, the PSK **MUST** be changed. Note that the PSK is different from the link-layer keys K1 and K2 specified in [RFC8180]. The PSK is a long-term secret used to protect the execution of the secure join protocol specified in this document; the link-layer keys are transported as part of the secure join protocol.

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 to a pledge is **RECOMMENDED**. However, due to operational constraints, the network identifier may not be known at the time of provisioning. If this parameter is not provisioned to the pledge, the pledge will attempt 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 provisioned, 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 the following:

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 [RFC8415], Generic Autonomic Signaling Protocol (GRASP) [RFC8990], or Multicast DNS (mDNS) [RFC6762]; the use of these mechanisms 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, telling 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 EB provides the link-layer 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. The advertisement 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 the 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:

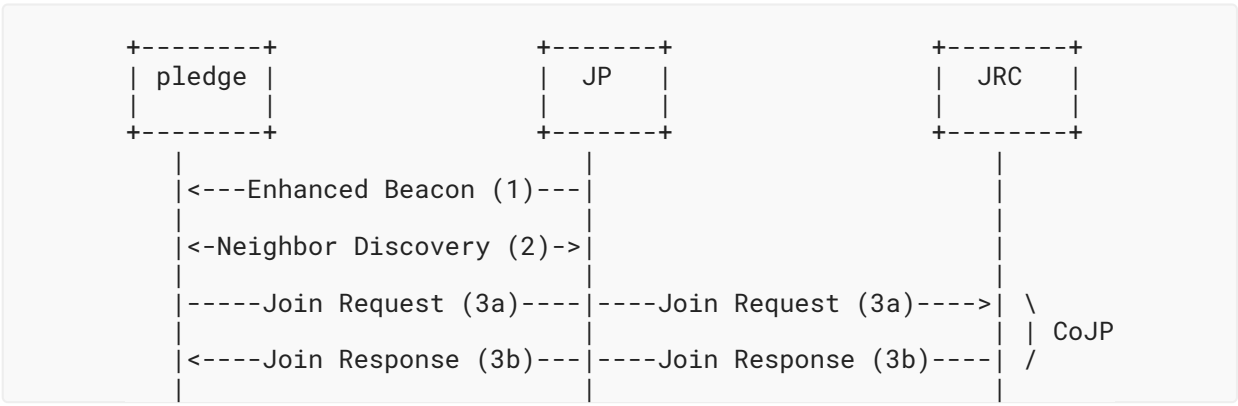

Figure 1: Overview of a successful join process.

As for 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 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 the following: which 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, at which signal strength the different EBs were received, etc. In addition, the pledge may be preconfigured 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 either 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 per [RFC4944]. The pledge **MAY** perform the Neighbor Solicitation / Neighbor Advertisement exchange with the JP per Section 5.6 of [RFC8505]. Per [RFC8505], there is no need to perform duplicate address detection for the link-local address. 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 the 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 (Section 4.3.1)) and the Join Response message, conditioned on the successful security processing of the request (Step 3b (Section 4.3.2)).

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 the 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 it 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 an application-layer proxy, see Section 7.

The Join Response contains various 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 just like 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 CoJP is out of scope of this document.

5. Link-Layer Configuration

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

Any node sending EB frames **MUST** be prepared to act as a JP for potential pledges.

The pledge does not initially perform an authenticity check of the EB frames because 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 at the link layer. In order for the join process to be possible, the JP must accept 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. It is expected that the lower layer provides an interface to indicate to the upper layer that unsecured frames are being received from a device. The upper layer can use that information to determine that a join process is in place and that the unsecured frames should be processed. How the JP makes such a determination and interacts with the lower layer is out of scope of this specification. The JP can additionally use information such as the value of the join rate parameter (Section 8.4.2) set by the JRC, physical button press, etc.

When the pledge initially synchronizes with the network, it has no means of verifying the authenticity of EB frames. Because an attacker can craft a frame that looks like a legitimate EB frame, this opens up a DoS vector, as discussed in Section 9.

5.1. Distribution of Time

Nodes in a 6TiSCH network keep a global notion of time known as the Absolute Slot Number. The Absolute Slot Number is used in the construction of the link-layer nonce, as defined in [IEEE802.15.4]. The pledge initially synchronizes with the EB frame sent by the JP and uses the value of the Absolute Slot Number found in the TSCH Synchronization Information Element. At the time of the synchronization, the EB frame can neither be authenticated nor its freshness verified. During the join process, the pledge sends frames that are unprotected at the link-layer and protected end-to-end instead. The pledge does not obtain the time information as the output of the join process as this information is local to the network and may not be known at the JRC.

This enables an attack on the pledge where the attacker replays to the pledge legitimate EB frames obtained from the network and acts as a man-in-the-middle between the pledge and the JP. The EB frames will make the pledge believe that the replayed Absolute Slot Number value is the current notion of time in the network. By forwarding the join traffic to the legitimate JP, the attacker enables the pledge to join the network. Under different conditions relating to the reuse of the pledge's short address by the JRC or its attempt to rejoin the network, this may cause the pledge to reuse the link-layer nonce in the first frame it sends protected after the join process is completed.

For this reason, all frames originated at the JP and destined to the pledge during the join process **MUST** be authenticated at the link layer using the key that is normally in use in the network. Link-layer security processing at the pledge for these frames will fail as the pledge is not yet in possession of the key. The pledge acknowledges these frames without link-layer security, and JP accepts the unsecured acknowledgment due to the secExempt attribute set for the pledge. The frames should be passed to the upper layer for processing using the promiscuous mode of [IEEE802.15.4] or another appropriate mechanism. When the upper-layer processing on the pledge is completed, and the link-layer keys are configured, the upper layer **MUST** trigger the security processing of the corresponding frame. Once the security processing of the frame carrying the Join Response message is successful, the current Absolute Slot Number kept locally at the pledge **SHALL** be declared as valid.

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 the 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 [RFC8505]. 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 may have an additional network interface that it uses in order to obtain the configuration parameters from the JRC and to 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 DODAG Information Objects (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 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 6TiSCH minimal network [RFC8180] 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 JP is aware of what traffic originates from unauthenticated pledges, and so can avoid allocating additional bandwidth itself. The JP implements a data cap on outgoing join traffic by implementing the recommendation of 1 packet per 3 seconds in Section 3.1.3 of [RFC8085]. This can be achieved with the congestion control mechanism specified in Section 4.7 of [RFC7252]. This cap will not protect intermediate nodes as they cannot tell join traffic from regular traffic. Despite the data cap implemented separately on each JP, the aggregate join traffic from many JPs may cause intermediate nodes to decide to allocate additional cells. It is undesirable to do so in response to the traffic originated from 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 JP **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 JP that does not require a specific DSCP value on forwarded traffic 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 JP to the AF42 code point. AF42 has lower drop probability than AF43, giving this traffic priority in buffers over the traffic going towards the JRC.

The 6LBR links are often the most congested within a DODAG, 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. If the recommended behavior described in this section is not followed, the network may become prone to the attack discussed in Section 6.1.

7. Application-Layer Configuration

The CoJP join exchange in Figure 1 is carried over CoAP [RFC7252] and the secure channel provided by OSCORE [RFC8613]. 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 the CoAP requests that 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 DoS attack.

This DoS vector on the JP can be mitigated by making the JP act as a stateless CoAP proxy, where "state" encompasses the information related 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 [RFC8974] to transport the state object. The JRC and any other potential proxy on the JP-JRC path **MUST** support extended token lengths, as defined in [RFC8974]. 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 that it forwards into the network on behalf of unauthenticated pledges. When the JP is operating in a stateless manner, the security considerations from [RFC8974] apply, and 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 stateful proxy that stores the per-pledge state until the response is received or timed out, but this comes at a price of a DoS vector.

7.2. Recommended Settings

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

Name	Default Value
ACK_TIMEOUT	10 seconds
ACK_RANDOM_FACTOR	1.5
MAX_RETRANSMIT	4
NSTART	1
DEFAULT_LEISURE	5 seconds
PROBING_RATE	1 byte/second

Table 1: Recommended CoAP settings.

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 per Section 3 of [RFC8613].

- 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.
- The Algorithm **MUST** be set to the value from [RFC8152], agreed to 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 by the same mechanism used to provision the PSK. Default is HKDF SHA-256 [RFC5869].

Since the pledge's OSCORE Sender ID is the empty byte string, when constructing the OSCORE option, the pledge sets the 'kid' flag in the OSCORE flag bits but indicates a 0-length 'kid'. The pledge transports its pledge identifier within the 'kid context' field of the OSCORE option. The derivation in [RFC8613] results in OSCORE keys and a Common Initialization Vector (IV) for each side of the conversation. Nonces are constructed by XORing the Common IV with the current sequence number. For details on nonce and OSCORE option construction, refer to [RFC8613].

Implementations **MUST** ensure that multiple CoAP requests, including to different JRCs, are properly incrementing the sequence numbers, so that the same sequence number is never reused in distinct requests protected under the same PSK. 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. 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 the 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. Note that when the (6LBR) pledge and the JRC change roles between CoAP client and CoAP server, the same OSCORE security context as initially derived remains in use, and the derived parameters are unchanged, for example, Sender ID when sending and Recipient ID when receiving (see Section 3.1 of [RFC8613]). A (6LBR) pledge is expected to have exactly one OSCORE security context with the JRC.

7.3.1. Replay Window and Persistency

Both the (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 [RFC8613] is **RECOMMENDED**.

Implementations **MUST** ensure that mutable OSCORE context parameters (Sender Sequence Number, Replay Window) are stored in persistent memory. A technique detailed in Appendix B.1.1 of [RFC8613] that prevents reuse of sequence numbers **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 [RFC8613]. CoJP allows a (6LBR) pledge to request admission into a network managed by the JRC. It enables the JRC to configure the pledge with the necessary parameters. 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.

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.

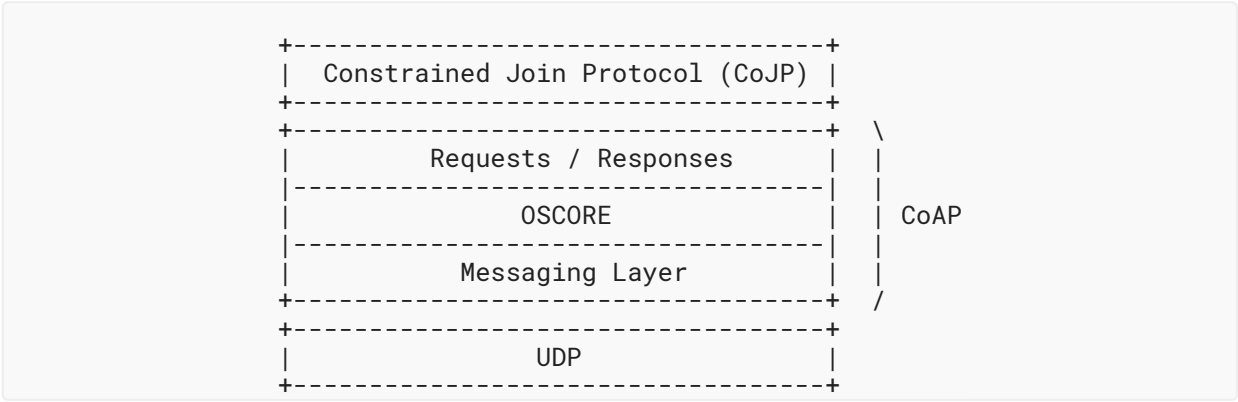

Figure 2: Abstract layering of CoJP.

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 8.1.1.
- 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 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 the following:

- 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.

The payload of CoJP messages is encoded with CBOR [RFC8949]. 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:

- The request method is POST.
- The type is Confirmable (CON).
- 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 OSCORE option **SHALL** be set according to [RFC8613]. 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.
- 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 implementation at the (6LBR) pledge declares the message transmission a failure, the (6LBR) pledge **SHOULD** attempt to join a 6TiSCH network advertised with a different network identifier. 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 the presence of an error condition, through some out-of-band mechanism.

BCP 190 [RFC8820] provides guidelines on URI design and ownership. It recommends that whenever a third party wants to mandate a URI to web authority that it **SHOULD** go under "/.well-known" (per [RFC8615]). In the case of CoJP, the Uri-Host option is always set to "6tisch.arpa", and based upon the recommendations in Section 1 of [RFC8820], it is asserted that this document is the owner of the CoJP service. As such, the concerns of [RFC8820] do not apply, and thus the Uri-Path is only "j".

8.1.2. 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 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 creates a CoAP service at the Uri-Host value of "6tisch.arpa", and 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:

- The request method is POST.
- The type is Confirmable (CON).
- The Uri-Host option is set to "6tisch.arpa".
- The Uri-Path option is set to "j".
- The OSCORE option **SHALL** be set according to [RFC8613]. 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.
- The payload is a Configuration CBOR object, as defined in Section 8.4.2.

The JRC has implicit knowledge of 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.

If 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 a 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 the security considerations on the reuse of assigned resources apply, as discussed in Section 9.

8.3. Error Handling

8.3.1. CoJP CBOR Object Processing

CoJP CBOR objects are transported within both CoAP requests and responses. This section describes handling the cases in which certain CoJP CBOR object parameters are not supported by the implementation or their processing fails. See Section 7.3.2 for the handling of errors that may be raised by the underlying OSCORE implementation.

When such a parameter is detected in a CoAP request (Join Request message, Parameter Update message), a Diagnostic Response message **MUST** be returned. A Diagnostic Response message maps to a CoAP response and is specified in Section 8.3.2.

When a parameter that cannot be acted upon is encountered 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 Unsupported Configuration CBOR object within the Join Request object in the following Join Request message. The Unsupported Configuration CBOR object is self-contained and enables the (6LBR) pledge to signal any parameters that the implementation of the networking stack may not support. A (6LBR) pledge **MUST NOT** attempt more than COJP_MAX_JOIN_ATTEMPTS number of attempts to join if the processing of the Join Response message fails each time. If the COJP_MAX_JOIN_ATTEMPTS number of attempts is reached without success, the (6LBR) pledge **SHOULD** signal the presence of an error condition through some out-of-band mechanism.

Note that COJP_MAX_JOIN_ATTEMPTS relates to the application-layer handling of the CoAP response and is different from CoAP's MAX_RETRANSMIT setting, which drives the retransmission mechanism of the underlying CoAP message.

8.3.2. Diagnostic Response Message

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

The Diagnostic Response message **SHALL** be mapped to a CoAP response:

- The Response Code is 4.00 (Bad Request).
- The payload is an Unsupported Configuration CBOR object, as defined in Section 8.4.5, containing more information about the parameter 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, a joined node or the JRC may 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). As the pledge (joined node) is typically a constrained device that handles the write operations to persistent memory in a predictable manner, the retrieval of mutable security-context parameters is feasible across reboots such that there is no risk of AEAD nonce reuse due to reinitialized Sender Sequence Numbers or of a replay attack due to the reinitialized Replay Window. The JRC may be hosted on a generic machine where the write operation to persistent memory may lead to unpredictable delays due to caching. If a reboot event occurs at the JRC before the cached data is written to persistent memory, the loss of mutable security-context parameters is likely, which consequently poses the risk of AEAD nonce reuse.

In the event of a complete device failure, where the mutable security-context parameters cannot be retrieved, it is expected that a failed joined node will be replaced with a new physical device, using a new pledge identifier and a PSK. When such a failure event occurs at the JRC, it is possible that the static information on provisioned pledges, like PSKs and pledge identifiers, can be retrieved through available backups. However, 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, the network administrator **MUST** force all the networks managed by the failed JRC to rejoin through out-of-band means during the process of JRC reinitialization, e.g., reinitialize the 6LBR nodes and freshly generate dynamic cryptographic keys and other parameters that influence the security properties of the network.

In order to recover from such a failure event, the reinitialized JRC can trigger the renegotiation of the OSCORE security context through the procedure described in Appendix B.2 of [RFC8613]. Aware of the failure event, the reinitialized JRC responds to the first Join Request of each pledge it is managing with a 4.01 (Unauthorized) error and a random nonce. The pledge verifies the error response and then initiates the CoJP join exchange using a new OSCORE security context derived from an ID Context consisting of the concatenation of two nonces, one that it received from the JRC and the other that the pledge generates locally. After verifying the Join Request with the new ID Context and the derived OSCORE security context, the JRC should consequently map the new ID Context to the previously used pledge identifier. How the JRC handles this mapping is out of scope of this document.

The use of the procedure specified in Appendix B.2 of [RFC8613] is **RECOMMENDED** in order to handle the failure events or any other event that may lead to the loss of mutable security-context parameters. The length of nonces exchanged using this procedure **MUST** be at least 8 bytes.

The procedure requires both the pledge and the JRC to have good sources of randomness. While this is typically not an issue at the JRC side, the constrained device hosting the pledge may pose limitations in this regard. If the procedure outlined in Appendix B.2 of [RFC8613] is not supported by the pledge, the network administrator **MUST** reprovision the concerned devices with freshly generated parameters through out-of-band means.

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 as part of 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 "Constrained Join Protocol (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 3. This parameter **MAY** be included. If 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 which network 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.

unsupported configuration: The identifier of the parameters that are not supported by the implementation, encoded as an Unsupported_Configuration 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 act on its payload (Configuration object), it **SHOULD** include this parameter to facilitate the recovery and debugging.

Table 2 summarizes the parameters that may appear in a Join_Request object.

Name	Label	CBOR Type
role	1	unsigned integer
network identifier	5	byte string
unsupported configuration	8	array

Table 2: Summary of Join_Request parameters.

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

```
Join_Request = {
    ? 1 : uint,                         ; role
      5 : bstr,                         ; network identifier
    ? 8 : Unsupported_Configuration     ; unsupported configuration
}
```

Name	Value	Description	Reference
6TiSCH Node	0	The pledge requests to play the role of a regular 6TiSCH node, i.e., non-6LBR node.	RFC 9031
6LBR	1	The pledge requests to play the role of 6LoWPAN Border Router (6LBR).	RFC 9031

Table 3: 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 "Constrained Join Protocol (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. This parameter is also used to implement rekeying in the network. The installation and use of keys differs for the 6LBR and other (regular) nodes, and this is explained in Sections 8.4.3.1 and 8.4.3.2.

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. Since the blacklist parameter carries the pledge identifiers, privacy considerations apply. See Section 10.

join rate: The average data rate (in units of bytes/second) 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. 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 to 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. If this parameter is set to zero, a joined node **MUST** silently drop any join traffic coming from unauthenticated pledges. If this parameter is omitted, the value of positive infinity **SHOULD** be assumed. A node operating as a JP **MAY** use another mechanism that is out of scope of this specification to configure the PROBING_RATE of CoAP in the absence of a join rate parameter from the Configuration object.

Table 4 summarizes the parameters that may appear in a Configuration object.

Name	Label	CBOR Type
link-layer key set	2	array
short identifier	3	array
JRC address	4	byte string
blacklist	6	array
join rate	7	unsigned integer

Table 4: Summary of Configuration parameters.

The CDDL fragment that represents the text above for the Configuration follows. The structures Link_Layer_Key and Short_Identifier are specified in Sections 8.4.3 and 8.4.4, respectively.

```
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	RFC 9031
link-layer key set	2	array	Identifies the array carrying one or more link-layer cryptographic keys	RFC 9031
short identifier	3	array	Identifies the assigned short identifier	RFC 9031
JRC address	4	byte string	Identifies the IPv6 address of the JRC	RFC 9031
network identifier	5	byte string	Identifies the network identifier parameter	RFC 9031
blacklist	6	array	Identifies the blacklist parameter	RFC 9031
join rate	7	unsigned integer	Identifier the join rate parameter	RFC 9031
unsupported configuration	8	array	Identifies the unsupported configuration parameter	RFC 9031

Table 5: 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 with which it should be used for both 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 [RFC8742], some being optional.

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

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. If the key is considered invalid, the key **MUST** be discarded, and the implementation **MUST** signal the error as specified in Section 8.3.1.

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 the IANA

"Constrained Join Protocol (CoJP) Key Usage" registry (Section 11.2). If the parameter is omitted, the default value of 0 (6TiSCH-K1K2-ENC-MIC32) from Table 6 **MUST** be assumed. This default value has been chosen because it results in byte savings in the most constrained settings; its selection does not imply a recommendation for its general usage.

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.

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 for the Link_Layer_Key that represents the text above follows:

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

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

Name	Value	Algorithm	Description
6TiSCH-K1-MIC32	6	IEEE802154-AES-CCM-128	Use MIC-32 for EBs.
6TiSCH-K1-MIC64	7	IEEE802154-AES-CCM-128	Use MIC-64 for EBs.
6TiSCH-K1-MIC128	8	IEEE802154-AES-CCM-128	Use MIC-128 for EBs.
6TiSCH-K2-MIC32	9	IEEE802154-AES-CCM-128	Use MIC-32 for DATA and ACKNOWLEDGMENT.
6TiSCH-K2-MIC64	10	IEEE802154-AES-CCM-128	Use MIC-64 for DATA and ACKNOWLEDGMENT.
6TiSCH-K2-MIC128	11	IEEE802154-AES-CCM-128	Use MIC-128 for DATA and ACKNOWLEDGMENT.
6TiSCH-K2-ENC-MIC32	12	IEEE802154-AES-CCM-128	Use ENC-MIC-32 for DATA and ACKNOWLEDGMENT.
6TiSCH-K2-ENC-MIC64	13	IEEE802154-AES-CCM-128	Use ENC-MIC-64 for DATA and ACKNOWLEDGMENT.
6TiSCH-K2-ENC-MIC128	14	IEEE802154-AES-CCM-128	Use ENC-MIC-128 for DATA and ACKNOWLEDGMENT.

Table 6: Key Usage values.

8.4.3.1. Rekeying of 6LBRs

When the 6LBR receives the Configuration object containing a link-layer key set, 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. This mechanism is used by the JRC to force the 6LBR to start sending traffic with the new key. The decision is made by the JRC when it has determined that the new key has been made available to all (or some overwhelming majority) of nodes. Any node that the JRC has not yet reached at that point is either nonfunctional or in extended sleep such that it will not be reached. To get the key update, such a node will need to go through the join process anew.

8.4.3.2. Rekeying of 6LNs

When a regular 6LN receives the Configuration object with a link-layer key set, it **MUST** install the new keys. The 6LN will use both the old and the new keys to decrypt and authenticate any incoming traffic that arrives based upon the key identifier in the packet. It **MUST** continue to use the old keys for all outgoing traffic until it has detected that the network has switched to the new key set.

The detection of the network switch is based upon the receipt of traffic secured with the new keys. Upon the reception and the successful security processing of a link-layer frame secured with a key from the new key set, a 6LN **MUST** then switch to sending all outgoing traffic using the keys from the new set. The 6LN **MUST** remove any keys it had installed from the previous key set after waiting COJP_REKEYING_GUARD_TIME since it started using the new key set.

Sending traffic with the new keys signals to other downstream nodes to switch to their new key, causing a ripple of key updates around each 6LBR.

8.4.3.3. 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. For instance, the value of the key_id parameter in combination with key_addinfo denotes which of the four Key ID modes of [IEEE802.15.4] is used and how.

Key ID Mode 0x00 (Implicit, pairwise): The key_id parameter **MUST** be set to 0. The key_addinfo parameter **MUST** be present. The 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 type(s) is carried is determined from the length of the byte string.

Key ID Mode 0x01 (Key Index): The key_id parameter **MUST** be set to a value different from 0. The key_addinfo parameter **MUST NOT** be present.

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

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

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

For Key ID Modes 0x01 through 0x03, the key_id parameter sets the "secKeyIndex" parameter of [IEEE802.15.4] that is signaled in all outgoing frames secured with a given key. The maximum value that 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 the JRC learns about the network topology is out of scope of this specification, but it 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.

Implementations **MUST** use different link-layer keys when using different authentication tag (MIC) lengths, as using the same key with different authentication tag lengths might be unsafe. For example, this prohibits the usage of the same key for both MIC-32 and MIC-64 levels. See Annex B.4.3 of [IEEE802.15.4] for more information.

8.4.4. Short Identifier

The Short_Identifier object represents an identifier assigned to the pledge. It is encoded as a CBOR array object and contains, 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. If 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. If the lease expires, then the node **SHOULD** initiate a new join exchange, as described in Section 8.1. If this parameter is omitted, then 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 for the Short_Identifier that represents the text above follows:

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

8.4.4.1. Use in IEEE Std 802.15.4

When the 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 the 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 from 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 imminent. 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 of the 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.

Care needs to be taken on how the pledge (joined node) configures the expiration of the lease. Since units of the lease_time parameter are in hours after the reception of the CBOR object, the pledge needs to convert the received time to the corresponding Absolute Slot Number in the network. The joined node (pledge) **MUST** only use the Absolute Slot Number as the appropriate reference of time to determine whether the assigned short identifier is still valid.

8.4.5. Unsupported Configuration Object

The Unsupported_Configuration object is encoded as a CBOR array, containing at least one Unsupported_Parameter object. Each Unsupported_Parameter object is a sequence of CBOR elements without an enclosing top-level CBOR object for compactness. The set of parameters that appear in an Unsupported_Parameter object is summarized below, in order:

code: Indicates the capability of acting on the parameter signaled by parameter_label, encoded as an integer. This parameter **MUST** be included. Possible values of this parameter are specified in the IANA "Constrained Join Protocol (CoJP) Unsupported Configuration Codes" registry (Section 11.3).

parameter_label: Indicates the parameter. This parameter **MUST** be included. Possible values of this parameter are specified in the label column of the IANA "Constrained Join Protocol (CoJP) Parameters" registry" (Section 11.1).

parameter_addinfo: Additional information about the parameter that cannot be acted upon. This parameter **MUST** be included. If the code is set to "Unsupported", parameter_addinfo gives additional information to the JRC. If the parameter indicated by parameter_label cannot

be acted upon regardless of its value, parameter_addinfo **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again. If the pledge can act on the parameter, but cannot configure the setting indicated by the parameter value, the pledge can hint this to the JRC. In this case, parameter_addinfo **MUST** be set to the value of the parameter that cannot be acted upon following the normative parameter structure specified in this document. For example, it is possible to include the link-layer key set object, signaling that either a subset or the entire key set that was received cannot be acted upon. In that case, the value of parameter_addinfo follows the link-layer key set structure defined in Section 8.4.2. If the code is set to "Malformed", parameter_addinfo **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again.

The CDDL fragment for the Unsupported_Configuration and Unsupported_Parameter objects that represents the text above follows:

```
Unsupported_Configuration = [
       + parameter           : Unsupported_Parameter
]

Unsupported_Parameter = (
         code                : int,
         parameter_label     : int,
         parameter_addinfo   : nil / any
)
```

Name	Value	Description	Reference
Unsupported	0	The indicated setting is not supported by the networking stack implementation.	RFC 9031
Malformed	1	The indicated parameter value is malformed.	RFC 9031

Table 7: Unsupported Configuration code values.

8.5. Recommended Settings

This section gives **RECOMMENDED** values of CoJP settings.

Name	Default Value
COJP_MAX_JOIN_ATTEMPTS	4
COJP_REKEYING_GUARD_TIME	12 seconds

Table 8: 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 the 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. While the process of provisioning PSKs to all pledges can result in a substantial operational overhead, it is vital to do so for the security properties of the network. 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. The resulting security context shared between the pledge (joined node) and the JRC is used for the purpose of joining and is long-lived in that it can be used throughout the lifetime of a joined node for parameter update exchanges. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible.

Note that while OSCORE provides replay protection, it does not provide an indication of freshness in the presence of an attacker that can drop and/or reorder traffic. Since the Join Request contains no randomness, and the sequence number is predictable, the JRC could in principle anticipate a Join Request from a particular pledge and pre-calculate the response. In such a scenario, the JRC does not have to be alive at the time the request is received. This could be relevant in the case when the JRC was temporarily compromised and control was subsequently regained by the legitimate owner.

It is of utmost importance to avoid 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. Additionally, some vendors use methods such as scrambling or hashing device serial numbers or their EUI-64 identifiers 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 and enables throttling in case of an attack. Note that this traffic can only be directed at the JRC so that the JRC needs to be prepared to handle such unsanitized inputs. 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 nonzero DSCP allows the network to carry the traffic if it has capacity, but it 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 the 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". If 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 the pledge fight a DoS attack. The bogus beacons prolong the join time of the pledge and so does the time spent in "minimal" duty cycle mode [RFC8180]. The blacklist communicated as part of the CoJP Configuration object helps the JP fight a DoS attack by a malicious pledge.

During the network lifetime, the JRC may at any time initiate a parameter update exchange with a joined node. The Parameter Update message uses the same OSCORE security context as is used for the join exchange, except that the server and client roles are interchanged. As a consequence, each Parameter Update message carries the well-known OSCORE Sender ID of the JRC. A passive attacker may use the OSCORE Sender ID to identify the Parameter Update traffic if the link-layer protection does not provide confidentiality. A countermeasure against such a traffic-analysis attack is to use encryption at the link layer. Note that the join traffic does not undergo link-layer protection at the first hop, as the pledge is not yet in possession of cryptographic keys. Similarly, EB traffic in the network is not encrypted. This makes it easy for a passive attacker to identify these types of traffic.

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 the 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. However, after the join process completes, the use of EUI-64 in the form of a Layer 2 or Layer 3 address extends the aforementioned privacy threats to the long term.

As an optional mitigation technique, the Join Response message may contain a short address that 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 may use the short addresses for all further Layer 2 (and Layer 3) operations. This reduces the privacy threats as the short Layer 2 address (visible even when the network is encrypted) 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

This document allocates a well-known name under the .arpa name space according to the rules given in [RFC3172] and [RFC6761]. The name "6tisch.arpa" is requested. No subdomains are expected, and addition of any such subdomains requires the publication of an IETF Standards Track RFC. No A, AAAA, or PTR record is requested.

11.1. Constrained Join Protocol (CoJP) Parameters

This section defines a subregistry within the "IPv6 Over the TSCH Mode of IEEE 802.15.4 (6TiSCH)" registry with the name "Constrained Join Protocol (CoJP) Parameters".

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. The name **MUST** be unique.

Label: The value to be used to identify this parameter. The label is an integer. The label **MUST** be unique.

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

Description: This field contains a brief description for the field. The description **MUST** be unique.

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

This registry is populated with the values in Table 5.

The amending formula for this subregistry 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. Constrained Join Protocol (CoJP) Key Usage

This section defines a subregistry within the "IPv6 Over the TSCH Mode of IEEE 802.15.4 (6TiSCH)" registry with the name "Constrained Join Protocol (CoJP) Key Usage".

The columns of this registry are:

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

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. The algorithm **MUST** be unique.

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. The description **MUST** be unique.

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

This registry is populated with the values in Table 6.

The amending formula for this subregistry 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. Constrained Join Protocol (CoJP) Unsupported Configuration Codes

This section defines a subregistry within the "IPv6 Over the TSCH Mode of IEEE 802.15.4 (6TiSCH)" registry with the name "Constrained Join Protocol (CoJP) Unsupported Configuration Codes".

The columns of this registry are:

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

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

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

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 7.

The amending formula for this subregistry 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. References

12.1. Normative References

[IEEE802.15.4] IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Standard 802.15.4-2015, DOI 10.1109/IEEESTD.2016.7460875, April 2016, <https://ieeexplore.ieee.org/document/7460875>.

[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>.

[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>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[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>.

[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>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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>.

[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>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[RFC8820] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 8820, DOI 10.17487/RFC8820, June 2020, <https://www.rfc-editor.org/info/rfc8820>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8974] Hartke, K. and M. Richardson, "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", RFC 8974, DOI 10.17487/RFC8974, January 2021, <https://www.rfc-editor.org/info/rfc8974>.

[RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <https://www.rfc-editor.org/info/rfc9030>.

12.2. Informative References

[NIST800-90A] National Institute of Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", Special Publication 800-90A, Revision 1, DOI 10.6028/NIST.SP.800-90Ar1, June 2015, <https://doi.org/10.6028/NIST.SP.800-90Ar1>.

[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>.

[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>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[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>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

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 it uses 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 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 CoJP object (configuration) present in the payload. At the pledge, the Join Response is matched to the Join Request and verified for replay protection using OSCORE processing rules. In this example, the Join Response does not contain the IPv6 address of the JRC, hence the pledge understands that the JRC is co-located with the 6LBR.

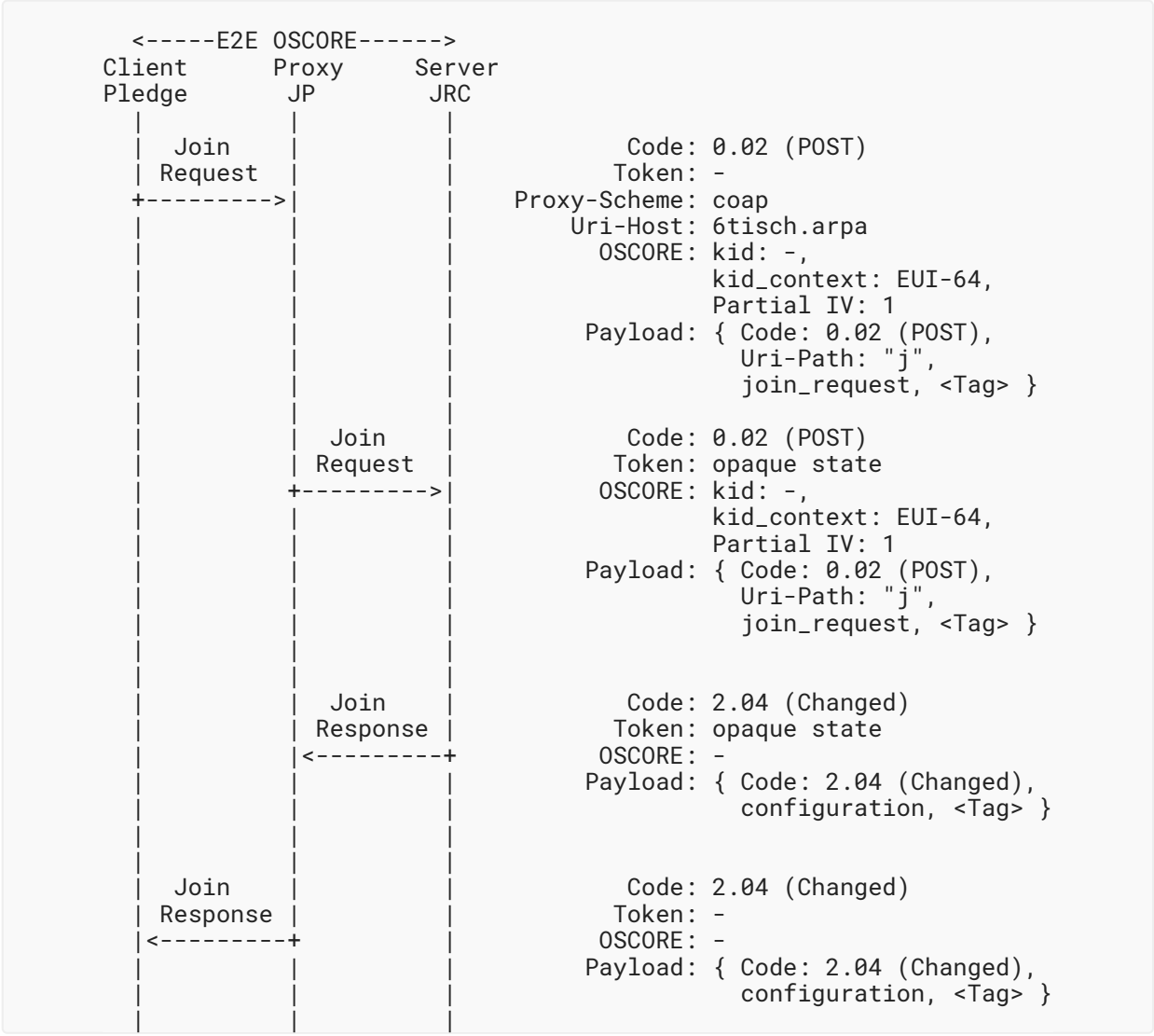

Figure 3: Example of a successful join protocol exchange. { ... } denotes authenticated encryption, <Tag> denotes the authentication tag.

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 is converted to h'a10542cafe' with a size of 5 bytes.

And the configuration object is the following:

```
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 the key_addinfo parameter is not present and key_id is different from 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 is converted to the following:

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 to 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 varies for different pledges. 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 a level of security comparable to the scenario in which each pledge is provisioned with a unique PSK. In this case, there is still a latent risk of the shared PSK being compromised on the provisioning device, which would compromise all devices in the batch.

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): Christian Amsüss, Tengfei Chang, Roman Danyliw, Linda Dunbar, Vijay Gurbani, Klaus Hartke, Barry Leiba, Benjamin Kaduk, Tero Kivinen, Mirja Kühlewind, John Mattsson, Hilarie Orman, Alvaro Retana, Adam Roach, Jim Schaad, Göran Selander, Yasuyuki Tanaka, Pascal Thubert, William Vignat, Xavier Vilajosana, Éric Vyncke, and Thomas Watteyne.

Authors' Addresses

Mališa Vučinić (EDITOR)
Inria
2 Rue Simone Iff
75012 Paris
France
Email: malisa.vucinic@inria.fr

Jonathan Simon
Analog Devices
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
United States of America
Email: jonathan.simon@analog.com

Kris Pister
University of California Berkeley
512 Cory Hall
Berkeley, CA 94720
United States of America
Email: pister@eecs.berkeley.edu

Michael Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa ON K1Z5V7
Canada
Email: mcr+ietf@sandelman.ca