6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

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

October 30, 2017

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-04

Abstract

This document describes the minimal configuration required for a new device, called "pledge", to securely join a 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) network. The entities involved use CoAP (Constrained Application Protocol) and OSCORE (Object Security for Constrained RESTful Environments). The configuration 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. The result of the joining process is that the JRC configures the pledge with link-layer keying material and a short link-layer address. This specification also defines a new Stateless-Proxy CoAP option. Additional security mechanisms may be added on top of this minimal framework.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

This document defines a secure join solution for a new device, called "pledge", to securely join a 6TiSCH network. The specification configures different layers of the 6TiSCH protocol stack and also defines a new CoAP option. It assumes the presence of a JRC (join registrar/coordinator), a central entity. It further assumes that the pledge and the JRC share a symmetric key, called PSK (pre-shared key). How the PSK is installed is out of scope of this document.

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These

words may also appear in this document in lowercase, absent their normative meanings.

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-terminology], [RFC7252], [I-D.ietf-core-object-security], and [RFC8152].

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

The following terms are used throughout this document:

pledge: The new device that wishes to join a 6TiSCH network.

joined node: The new device, after having completed the join process, often just called a node.

join proxy (JP): A node already part of the 6TiSCH network that serves as a relay to provide connectivity between the pledge and the JRC.

join registrar/coordinator (JRC): A central entity responsible for the authentication, authorization and configuration of the pledge.

3. One-Touch Assumption

This document assumes a one-touch scenario. The pledge is provisioned with a PSK before attempting to join the network, and the same PSK (as well as the uniquer identifier of the pledge) is provisioned on the JRC.

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

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

3.1. Pre-Shared Key

The PSK SHOULD be at least 128 bits in length, generated uniformly at random. It is RECOMMENDED to generate the PSK with a

cryptographically secure pseudorandom number generator. Each pledge SHOULD be provisioned with a unique PSK.

4. Join Overview

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

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE802.15.4-2015]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which plays the role of join proxy (JP) for the pledge, and when it can expect to receive a frame.

2. The pledge configures its link-local IPv6 address and advertises it to the join proxy (JP).

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

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

From the pledge's perspective, minimal joining is a local phenomenon - the pledge only interacts with the JP, and it need 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 process is shown as a transaction diagram in Figure 1:

```
+--------+                 +-------+                 +--------+
| pledge |                 |  JP   |                 |  JRC   |
|        |                 |       |                 |        |
+--------+                 +-------+                 +--------+
   |                          |                          |
   |<---Enhanced Beacon (1)---|                          |
   |                          |                          |
   |<-Neighbor Discovery (2)->|                          |
   |                          |                          |
   |-----Join Request (3)-----|------Join Request (3a)-->|
   |                          |                          |
   |<---Join Response (4)-----|-----Join Response (4a)---|
   |                          |                          |
```

Figure 1: Overview of a successful join process.

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

4.1. Step 1 - Enhanced Beacon

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

Once the pledge hears an EB, it synchronizes to the joining schedule using the cells contained in the EB. The pledge can hear multiple EBs; the selection of which EB to use is out of the scope for this document, and is discussed in [RFC7554]. Implementers SHOULD make use of information such as: what Personal Area Network Identifier (PAN ID) [IEEE802.15.4-2015] the EB contains, whether the source link-layer address of the EB has been tried before, what signal strength the different EBs were received at, etc. In addition, the pledge may be pre-configured to search for EBs with a specific PAN ID.

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

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

4.2. Step 2 - Neighbor Discovery

The pledge forms its link-local IPv6 address based on EUI-64, as per [RFC4944]. The Neighbor Discovery exchange shown in Figure 1 refers to a single round trip Neighbor Solicitation / Neighbor Advertisement exchange between the pledge and the JP (Section 5.5.1 of [RFC6775]). The pledge uses the link-local IPv6 address for all subsequent communication with the JP during the join process.

Note that ND exchanges at this point are not protected with link-layer security as the pledge is not in possession of the keys. How JP accepts these unprotected frames is discussed in Section 12.

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

4.3. Step 3 - Join Request

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

The Join Request is authenticated/encrypted end-to-end using an AEAD algorithm from [RFC8152] and a key derived from the PSK, the pledge's EUI-64 and a request-specific constant value. Algorithms which MUST be implemented are specified in Section 11.

The nonce used when securing the Join Request is derived from the PSK, the pledge's EUI-64 and a monotonically increasing counter initialized to 0 when first starting.

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

4.4. Step 4 - Join Response

The Join Response is sent by the JRC to the pledge, and is forwarded through the JP as it serves as a stateless relay. The packet containing the Join Response travels from the JRC to JP using the operating routes in the 6TiSCH network. The JP delivers it to the pledge using the slot information it has indicated in the EB it sent. The JP operates as the application-layer proxy, and does not keep any state to relay the message. It uses information sent in the clear within the Join Response to decide where to forward to.

The Join Response is authenticated/encrypted end-to-end using an AEAD algorithm from [RFC8152]. The key used to protect the response is different from the one used to protect the request (both are derived from the PSK, as explained in Section 6). The response is protected using the same nonce as in the request.

The Join Response contains one or more link-layer key(s) that the pledge will use for subsequent communication. Each key that is provided by the JRC is associated with an 802.15.4 key identifier. In other link-layer technologies, a different identifier may be substituted. The Join Response also contains an IEEE 802.15.4 short address [IEEE802.15.4-2015] assigned by the JRC to the pledge, and optionally the IPv6 address of the JRC.

Join Response construction is specified in Section 8, while the details on processing can be found in Section 7 of [I-D.ietf-core-object-security].

5. Architectural Overview and Communication through Join Proxy

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

The pledge communicates with a JP over link-local IPv6 addresses. The pledge designates a JP as a proxy by including the Proxy-Scheme option with value "coap" (CoAP-to-CoAP proxy) in CoAP requests it sends to the JP. The pledge MUST include the Uri-Host option with its value set to the well-known JRC's alias "6tisch.arpa". This allows the pledge to join without knowing the IPv6 address of the JRC. The pledge learns the actual IPv6 address of the JRC from the Join Response; it uses it once joined in order to operate as a JP.

The JRC can be co-located on the 6LBR. Before the 6TiSCH network is started, the 6LBR MUST be provisioned with the IPv6 address of the JRC.

5.1. Stateless-Proxy CoAP Option

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

The Stateless-Proxy CoAP option Figure 2 allows the JP to be entirely stateless. This option inserts, in the request, the state information needed for relaying the response back to the client. The proxy still keeps some general state (e.g. for congestion control or request retransmission), but no per-client state.

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

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

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

Figure 2: Stateless-Proxy CoAP Option

Upon reception of a Stateless-Proxy option, the CoAP server MUST echo it in the response. The value of the Stateless-Proxy option is internal proxy state that is opaque to the server. Example state information includes the IPv6 address of the client, its UDP source port, and the CoAP token. For security reasons, the state information MUST be authenticated, MUST include a freshness indicator (e.g. a sequence number or timestamp) and MAY be encrypted. The proxy may use an appropriate COSE structure [RFC8152] to wrap the state information as the value of the Stateless-Proxy option. The key used for encryption/authentication of the state information may be known only to the proxy.

Once the proxy has received the CoAP response with Stateless-Proxy option present, it decrypts/authenticates it, checks the freshness

indicator and constructs the response for the client, based on the information present in the option value.

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

6. OSCORE Security Context

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

- the Master Secret MUST be the PSK.

- the Master Salt MUST be pledge's EUI-64.

- the Sender ID of the pledge MUST be set to byte string 0x00.

- the Recipient ID (ID of the JRC) MUST be set to byte string 0x01.

- the Algorithm MUST be set to the value from [RFC8152], agreed out-of-band by the same mechanism used to provision the PSK. The default is AES-CCM-16-64-128.

- the Key derivation function MUST be agreed out-of-band. Default is HKDF SHA-256.

The derivation in [I-D.ietf-core-object-security] results in traffic keys and a common IV for each side of the conversation. Nonces are constructed by XOR'ing the common IV with the current sequence number and sender identifier. For details on nonce construction, refer to [I-D.ietf-core-object-security].

It is RECOMMENDED that a PAN ID be provisioned to the pledge out-of-band by the same mechanism used to provision the PSK. This prevents the pledge from attempting to join a wrong network. If the pledge is not provisioned with the PAN ID, it SHOULD attempt to join one network at a time. In that case, implementations MUST ensure that multiple CoAP requests to different JRCs result in the use of the same OSCORE context so that sequence numbers are properly incremented for each request.

6.1. Persistency

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

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

7. Specification of Join Request

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

o The request method is POST.

o The type is Non-confirmable (NON).

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

o The Uri-Host option is set to "6tisch.arpa".

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

o The Object-Security option SHALL be set according to [I-D.ietf-core-object-security]. The OSCORE Context Hint SHALL be set to pledge's EUI-64. The OSCORE Context Hint allows the JRC to retrieve the security context for a given pledge.

o The payload is empty.

8. Specification of Join Response

If the JRC successfully processes the Join Request using OSCORE, and if the pledge is authorized to join the network, the Join Response the JRC sends back to the pledge SHALL be mapped to a CoAP response:

o The response Code is 2.04 (Changed).

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

   * the COSE Key Set, specified in [RFC8152], containing one or more link-layer keys. The mapping of individual keys to 802.15.4-specific parameters is described in Section 8.1.

* the link-layer short address to be used by the pledge. The format of the short address follows Section 8.2.

* optionally, the IPv6 address of the JRC transported as a byte string. If the IPv6 address of the JRC is not present in the Join Response, this indicates the JRC is co-located with 6LBR, and has the same IPv6 address as the 6LBR. The address of the 6LBR can then be learned from DODAGID field in RPL DIOs [RFC6550].

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

8.1. Link-layer Keys Transported in COSE Key Set

Each key in the COSE Key Set [RFC8152] SHALL be a symmetric key. If the "kid" parameter of the COSE Key structure is present, the corresponding keys SHALL belong to an IEEE 802.15.4 KeyIdMode 0x01 class. In that case, parameter "kid" of the COSE Key structure SHALL be used to carry the IEEE 802.15.4 KeyIndex value. If the "kid" parameter is not present in the transported key, the application SHALL consider the key to be an IEEE 802.15.4 KeyIdMode 0x00 (implicit) key. This document does not support IEEE 802.15.4 KeyIdMode 0x02 and 0x03 class keys.

8.2. Short Address

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

o Byte string, containing the 16-bit address.

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

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

It is up to the joined node to request a new short address before the expiry of its previous address. The mechanism by which the node requests renewal is the same as during join procedure, as described in Section 13. The assigned short address is used for configuring both link-layer short address and IPv6 addresses.

9. Error Handling and Retransmission

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

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

The following binary exponential back-off algorithm is inspired by the one described in [RFC7252]. For each Join Request the pledge sends while waiting for a Join Response, the pledge MUST keep track of a timeout and a retransmission counter. For a new Join Request, the timeout is set to a random value between TIMEOUT and (TIMEOUT * TIMEOUT_RANDOM_FACTOR), and the retransmission counter is set to 0. When the timeout is triggered and the retransmission counter is less than MAX_RETRANSMIT, the Join Request is retransmitted, the retransmission counter is incremented, and the timeout is doubled. Note that the retransmitted Join Request passes new OSCORE processing, such that the sequence number in the OSCORE context is properly incremented. If the retransmission counter reaches MAX_RETRANSMIT on a timeout, the pledge SHOULD attempt to join the next advertised 6TiSCH network. If the pledge receives a Join Response that successfully passed OSCORE processing, it cancels the pending timeout and processes the response. The pledge MUST silently discard any response not protected with OSCORE, including error codes. For default values of retransmission parameters, see Section 10.

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

10. Parameters

This specification uses the following parameters:

Name	Default Value
TIMEOUT	10 s
TIMEOUT_RANDOM_FACTOR	1.5
MAX_RETRANSMIT	4

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

11. Mandatory to Implement Algorithms

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

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

12. Link-layer Requirements

In an operational 6TiSCH network, all frames MUST use link-layer frame security [RFC8180]. The frame security options MUST include frame authentication, and MAY include frame encryption.

The pledge does not initially do any authentication of the EB frames, as it does not know the K1 key [RFC8180]. When sending frames, the pledge sends unencrypted and unauthenticated frames. The JP accepts these frames (using the "exempt mode" in 802.15.4) for the duration of the join process. How the JP learns whether the join process is ongoing is out of scope of this specification.

As the EB itself cannot be authenticated by the pledge, an attacker may craft a frame that appears to be a valid EB, since the pledge can neither know the ASN a priori nor verify the address of the JP. This opens up a possibility of DoS attack, as discussed in Section 14. Beacon authentication keys are discussed in [RFC8180].

13. Rekeying and Rejoin

This specification handles initial keying of the pledge. For reasons such as rejoining after a long sleep, expiry of the short address, or node-initiated rekeying, the joined node MAY send a new Join Request using the already-established OSCORE security context. The JRC then responds with up-to-date keys and a (possibly new) short address. How the joined node decides when to rekey is out of scope of this document. Mechanisms for rekeying the network are defined in companion specifications, such as [I-D.richardson-6tisch-minimal-rekey].

14. Security Considerations

This document recommends that the pledge and JRC are provisioned with unique PSKs. The request nonce and the response nonce are the same, but used under a different key. The design differentiates between keys derived for requests and keys derived for responses by different sender identifiers (0x00 for pledge and 0x01 for JRC). Note that the address of the JRC does not take part in nonce or key construction. Even in case of a misconfiguration in which the same PSK is used for several nodes, the keys used to protect the requests/responses from/towards different pledges are different, as they are derived using the pledge's EUI-64 as Master Salt. The PSK is still important for mutual authentication of the pledge and JRC. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible. The well-known problem with Bluetooth headsets with a "0000" pin applies here.

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

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

At the beginning of the join process, the pledge has no means of verifying the content in the EB, and has to accept it at "face value". In case the pledge tries to join an attacker's network, the Join Response message will either fail the security check or time out. The pledge may implement a blacklist in order to filter out undesired EBs and try to join using the next seemingly valid EB. This blacklist alleviates the issue, but is effectively limited by the node's available memory. Bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode.

15. Privacy Considerations

This specification relies on the uniqueness of the node's EUI-64 that is transferred in clear as an OSCORE Context Hint. Privacy implications of using such long-term identifier are discussed in [RFC7721] and comprise correlation of activities over time, location tracking, address scanning and device-specific vulnerability exploitation. Since the join protocol is executed rarely compared to the network lifetime, long-term threats that arise from using EUI-64 are minimal. In addition, the Join Response message contains a short address which is assigned by JRC to the pledge. The assigned short address SHOULD be uncorrelated with the long-term EUI-64 identifier. The short address is encrypted in the response. Use of short addresses once the join protocol completes mitigates the aforementioned privacy risks.

16. IANA Considerations

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

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

16.1. CoAP Option Numbers Registry

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

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

17. Acknowledgments

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

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

18. References

18.1. Normative References

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-06 (work in progress), October 2017.

[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>.

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

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

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

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

18.2. Informative References

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

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-09 (work in progress), June 2017.

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-ietf-cbor-cddl-00 (work in progress), July 2017.

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

[IEEE802.15.4-2015]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015.

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

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

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

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

[RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <https://www.rfc-editor.org/info/rfc7554>.

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

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

Appendix A. Example

Figure 3 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the traffic keys and nonces from the PSK. It uses the instantiated context to protect the Join Request addressed with a Proxy-Scheme option, the well-known host name of the JRC in the Uri-Host option, and its EUI-64 identifier as OSCORE Context Hint. Triggered by the presence of Proxy-Scheme option, the JP forwards the request to the JRC and adds the Stateless-Proxy option with value set to the internally needed state, authentication tag, and a freshness indicator. The JP learned the IPv6 address of JRC when it acted as a pledge and joined the network. Once the JRC receives the request, it looks up the correct context based on the Context Hint parameter. It reconstructs OSCORE's external Additional Authenticated Data (AAD) needed for verification based on:

- o the Version of the received CoAP header.
- o the Algorithm value agreed out-of-band, default being AES-CCM-16-64-128 from [RFC8152].
- o the Request ID being set to the value of the "kid" field of the received COSE object.

- the Join Request sequence number set to the value of "Partial IV" field of the received COSE object.

- Integrity-protected options received as part of the request.

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

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

Figure 3: Example of a successful join protocol exchange. { ... } denotes encryption and authentication, [...] denotes authentication.

Where join_response is as follows.

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

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

Authors' Addresses

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

Email: malisav@ac.me

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

Email: jonathan.simon@analog.com

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

Email: pister@eecs.berkeley.edu

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

Email: mcr+ietf@sandelman.ca