6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 6, 2017

M. Vucinic
Inria
J. Simon
Linear Technology
K. Pister
University of California Berkeley

February 02, 2017

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-01

Abstract

This draft describes the minimal mechanisms required to support secure initial configuration in a device being added to a 6TiSCH network. The goal of this configuration is to set link-layer keys, and to establish a secure session between each joining node and the JCE who may use that to further configure the joining device. Additional security behaviors and mechanisms may be added on top of this minimal framework.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 6, 2017.

Copyright Notice

Table of Contents

1. Introduction

When a previously unknown device seeks admission to a 6TiSCH [RFC7554] network (to "join"), it first needs to synchronize to the network. The device then configures its IPv6 address and authenticates itself, and also validates that it is joining the right network. At this point it can expect to interact with the network to configure its link-layer keying material. Only then may the node establish an end-to-end secure session with an Internet host using DTLS [RFC6347] or OSCOAP [I-D.ietf-core-object-security]. Once the

application requirements are known, the device interacts with its peers to request additional resources as needed, or to be reconfigured as the network changes [I-D.ietf-6tisch-6top-protocol].

This document describes the mechanisms comprising a minimal feature set for a device to join a 6TiSCH network, up to the point where it can establish a secure session with an Internet host.

It presumes a network as described by [RFC7554], [I-D.ietf-6tisch-6top-protocol], and [I-D.ietf-6tisch-terminology]. It assumes the joining device pre-configured with either a:

- pre-shared key (PSK),
- raw public key (RPK),
- or a locally-valid certificate and a trust anchor.

As the outcome of the join process, the joining device expects one or more link-layer key(s) and optionally a temporary network identifier.

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], and [I-D.ietf-core-object-security]. The entities participating in the protocol that is specified in this document are:

- JN: Joining node - the device attempting to join a particular 6TiSCH network.
- JCE: Join coordinating entity - central entity responsible for authentication and authorization of joining nodes.
- JA: Join assistant - the device within radio range of the JN that generates Enhanced Beacons (EBs) and facilitates end-to-end communications between the JN and JCE.

3. Join Overview

This section describes the steps taken by a joining node (JN) in a 6TiSCH network. When a previously unknown device seeks admission to a 6TiSCH [RFC7554] network, the following exchange occurs:

1. The JN listens for an Enhanced Beacon (EB) frame [IEEE802154-2015]. 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 Assistant (JA) for the JN, and when it can expect to receive a frame.

2. The JN configures its link-local IPv6 address and advertises it to JA.

3. The JN sends packets to the JA device in order to securely identify itself to the network. These packets are directed to the Join Coordination Entity (JCE), which may be the JA or another device.

4. The JN receives one or more packets from JCE (via the JA) that sets up one or more link-layer keys used to authenticate subsequent transmissions to peers.

From the joining node's perspective, minimal joining is a local phenomenon - the JN only interacts with the JA, and it need not know how far it is from the DAG root, or how to route to the JCE. Only after establishing one or more link-layer keys does it need to know about the particulars of a 6TiSCH network.

The handshake is shown as a transaction diagram in Figure 1:

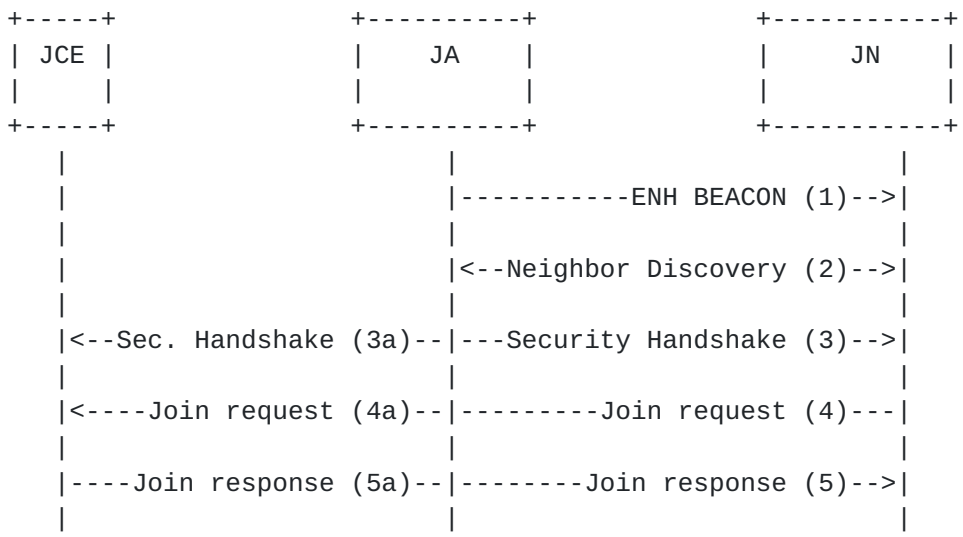

Figure 1: Message sequence for join protocol.

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

3.1. Step 1 - Enhanced Beacon

The JN hears an EB from the JA and synchronizes itself to the joining schedule using the cells contained in the EB. At this point the JN MAY proceed to step 2, or continue to listen for additional EBs. If more than one EB is heard, the JN MAY use a metric based on DAG rank and received signal level of the EB, or other factors to decide which JA to use for the security handshake in step 3. Details on how a JN chooses the JA are out of scope of this specification.

3.2. Step 2 - Neighbor Discovery

At this point, JN forms its link-local IPv6 address based on EUI64 and MAY further follow the Neighbor Discovery (ND) process described in Section 5 of [RFC6775].

3.3. Step 3 - Security Handshake

The security handshake between JN and JCE uses Ephemeral Diffie-Hellman over COSE (EDHOC) [I-D.selander-ace-cose-ecdhe] to establish the shared secret used to encrypt the join request and join response.

The security handshake step is OPTIONAL in case PSKs are used, while it is REQUIRED for RPKs and certificates. In case the handshake step is omitted, the shared secret used for protection of the join request and join response in the next step is the PSK. This means that the protocol trades off perfect forward secrecy for reduced traffic load between JN and JCE. A consequence is that if the long-term PSK is

compromised, keying material transferred as part of the join response is compromised as well. Physical compromise of the JN, however, would also imply the compromise of the same keying material, as it is likely to be found in node's memory.

3.3.1. Pre-Shared Key

The Diffie-Hellman key exchange and the use of EDHOC is optional, when using a pre-shared symmetric key. This cuts down on traffic between JCE and JN, but requires pre-configuration of the shared key on both devices.

It is REQUIRED to use unique PSKs for each JN.

3.3.2. Asymmetric Keys

The Security Handshake step is required, when using asymmetric keys. Before conducting the Diffie-Hellman key exchange using EDHOC [I-D.selander-ace-cose-ecdhe] the JN and JCE need to receive and validate each other's public key certificate. When RPKs are pre-configured at JN and JCE, they can directly proceed to the handshake.

3.4. Step 4 - Join Request

The join request is sent from the JN to the JA using the slot information from the EB, and forwarded to the JCE.

The join request is authenticated/encrypted end-to-end using AES-CCM-16-64-128 algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3. The nonce is derived from the shared secret, JN's EUI64 and a monotonically increasing counter initialized to 0 when first starting.

3.5. Step 5 - Join Response

The join response is sent from the JCE to the JN through JA that serves as a stateless relay. Packet containing the join response travels on the path from JCE to JA using pre-established routes in the network. The JA delivers it to the JN using the slot information from the EB. JA 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 using AES-CCM-16-64-128 algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3. The nonce is derived from the shared secret, JN's EUI64 and a monotonically increasing counter matching that of the join request.

The join response contains one or more (per-peer) link-layer key(s) K2 that the JN will use for subsequent communication. It optionally also contains an IEEE 802.15.4 short-address [IEEE802154-2015] assigned to JN by JCE.

4. Protocol Specification

The join protocol in Figure 1 is implemented over Constrained Application Protocol (CoAP) [RFC7252]. JN plays the role of a CoAP client, JCE the role of a CoAP server, while JA implements CoAP forward proxy functionality [RFC7252]. Since JA is likely a constrained device, it does not need to implement a cache but rather process forwarding-related CoAP options and make requests on behalf of JN that is not yet part of the network.

JN and JCE MUST protect their exchange end-to-end (i.e. through the proxy) using Object Security of CoAP (OSCOAP) [I-D.ietf-core-object-security].

4.1. Proxy Operation of JA

JN designates a JA as a proxy by including in the CoAP requests to the JA the Proxy-Scheme option with value "coap" (CoAP-to-CoAP proxy). JN MUST include the Uri-Host option with its value set to the well-known JCE's alias - "6tisch.jce". JN does not need to learn the actual IPv6 address of JCE at any time during the join protocol. JA resolves the address by performing a GET request at "/jce" resource of its parent in the DODAG.

Note that the CoAP proxy by default keeps state information in order to forward the response towards the originator of the request. This state information comprises CoAP token, but the implementations also need to keep track of the IPv6 address of the host, as well as the corresponding UDP source port number. In the setting where the proxy is a constrained device, as in the case of JA, this makes it prone to Denial of Service (DoS) attacks, due to the limited memory.

In order to facilitate a stateless implementation of JA proxying, JN shall encode in the CoAP message the information necessary for the JA to send the response back - "origin_info". For this purpose, JN uses the "Context Identifier (Cid)" parameter of OSCOAP's security context structure. Context Identifier is sent in clear, readable by JA, and MUST be echoed back in the response from JCE. This makes it possible to implement JA's CoAP proxy in a stateless manner. It also allows JCE to look up the right security context for communication with a given JN.

4.1.1. Implementation of origin_info

The origin_info is implemented as a CBOR [RFC7049] array object containing:

- EUI64: JN's EUI64 address
- source_port: JN's UDP source port
- token: JN's CoAP token

```
origin_info = [
    EUI64 : bstr,
    source_port : uint,
    token : uint
]
```

4.2. OSCOAP Security Context Instantiation

The OSCOAP security context MUST be derived at JN and JCE as per Section 3.2 of [I-D.ietf-core-object-security] using HKDF [RFC5869] as the key derivation function.

- Context Identifier (Cid) MUST be the origin_info object wrapped as a byte string (bstr).
- Algorithm MUST be set to AES-CCM-16-64-128 from [I-D.ietf-cose-msg]. CoAP messages are therefore protected with an 8-byte CCM authentication tag and the algorithm uses 13-byte long nonces.
- Base key (base_key) MUST be the secret generated by the run of EDHOC, or the PSK in case EDHOC step was omitted.
- Sender ID of JN MUST be set to 0x00, while the ID of JCE MUST be set to 0x01.

The hash algorithm that instantiates HKDF MUST be SHA-256 [RFC4231]. The derivation in [I-D.ietf-core-object-security] results in traffic keys and static IVs for each side of the conversation. Nonces are constructed by XOR'ing the static IV with current sequence number. The context derivation process occurs exactly once. Implementations MUST ensure that multiple CoAP requests to different JCEs result in the use of the same OSCOAP context so that sequence numbers are properly incremented for each request. This may happen in a scenario where there are multiple 6TiSCH networks present and the JN tries to join one network at a time.

4.3. Implementation of Join Request

Join Request message SHALL be mapped to a CoAP request:

- The request method is GET.
- The Proxy-Scheme option is set to "coap".
- The Uri-Host option is set to "6tisch.jce".
- The Uri-Path option is set to "j".
- The object security option SHALL be set according to [I-D.ietf-core-object-security] and OSCOAP parameters set as described above.

4.4. Implementation of Join Response

If OSCOAP processing is a success, Join Response message SHALL be a CoAP response:

- The response Code is 2.05 (Content).
- The payload is a CBOR array containing, in order:
  - COSE Key Set [I-D.ietf-cose-msg]. Each key in the Key Set SHALL be a symmetric key. A key that is present in the Key Set and does not have an identifier is assumed to be "K2" link-layer key from [I-D.ietf-6tisch-minimal]. Parameter "kid" of the COSE Key structure SHALL be used to denote pair-wise keys if present, where the value SHALL be set to the address of the corresponding peer.
  - Optional byte string representing IEEE 802.15.4 short address assigned to JN. If the length of the byte string is different than 2 bytes, the implementation SHOULD ignore it.

```
payload = [
    COSE_KeySet,
    ? short_address : bstr,
]
```

In case JCE determines that JN is not supposed to join the network (e.g. by failing to find an appropriate security context), it should respond with a 4.01 Unauthorized error. Upon reception of a 4.01 Unauthorized, JN SHALL attempt to join the next advertised 6TiSCH network. If all join attempts have failed at JN, JN SHOULD signal to

the user by an out-of-band mechanism the presence of an error condition.

5. Link-layer requirements

All frames in a 6TiSCH network MUST use link-layer frame security. The frame security options MUST include frame authentication, and MAY include frame encryption.

In order for the JN to be able to validate that the Enhanced Beacon frame is coming from a 6TiSCH network, EB frames are authenticated at the link layer using CCM* per [IEEE802154-2015]. Link-layer frames are protected with a 16-byte key, and a 13-byte nonce constructed from current Absolute Slot Number (ASN) and the source (the JA for EBs) address, as shown in Figure 2:

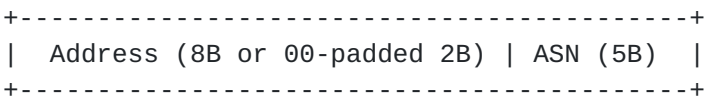

Figure 2: Link-layer CCM* nonce construction

The JN uses the initial key K1 [I-D.ietf-6tisch-minimal] until it is configured with a new link-layer key K2 as described above. JA SHOULD secure/verify DATA and ACKNOWLEDGMENT frames destined/originated at JN with K1 only during the duration of the join process. How JA learns whether the join process is ongoing is out of scope of this specification.

As the EB itself does not contain security information, where the link key is known, an attacker may craft a frame that appears to be a valid EB, since the JN can neither know the ASN a priori nor verify the address of the JA. This permits a Denial of Service (DoS) attack at the JN. Beacon authentication keys are discussed in Section 5.1 and Section 5.2.

5.1. Well-known beacon authentication key

For zero-touch operation, where any 6TiSCH device can attempt to join any 6TiSCH network out of the box, a well-known EB link-layer key MUST be used. The value of this key is specified in [I-D.ietf-6tisch-minimal]

5.2. Private beacon authentication key

Some pre-configuration MAY be done when the device is manufactured or designated for a specific network (i.e. the network is one-touch) or a network operator may not wish to allow arbitrary devices to try to

join. A private (per-vendor, or per-installation) EB link-layer key MAY be used in place of a well-known key to create a private network.

6. Asymmetric Keys

Certificates or pre-configured RPKs may be used to exchange public keys between the JN and JCE. The key pair is generated using elliptic curve secp256r1, and the certificate containing the public key is signed using ECDSA. The certificate itself may be a compact representation of an X.509 certificate, or a full X.509 certificate. Compact representation of X.509 certificates is out of scope of this specification. The certificate is signed by a root CA whose certificate is installed on all nodes participating in a particular 6TiSCH network, allowing each node to validate the certificate of the JCE or JN as appropriate.

7. Security Considerations

In case PSKs are used, this document mandates that JN and JCE are pre-configured with unique keys. The uniqueness of generated nonces is guaranteed under the assumption of unique EUI64 identifiers for each JN. Note that the address of the JCE does not take part in nonce construction. Therefore, even under the assumption of a PSK shared by a group of nodes, the nonces constructed as part of the different responses are unique. The design differentiates between nonces constructed for requests and nonces constructed for responses by different sender identifiers (0x00 for JN and 0x01 for JCE).

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

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

Because the well-known beacon authentication key does not provide any security, it is feasible for an attacker to generate EBs that will

get accepted at JN. At the time of the join, JN has no means of verifying the content in the EB and has to accept it at "face value". As the join response message in such cases will either fail the security check or time out, JN may implement a blacklist in order to filter out undesired beacons and try to join the next seemingly valid network. The blacklist alleviates the issue but is effectively limited by the node's available memory. Such bogus beacons will prolong the join time of JN and so the time spent in "minimal" [I-D.ietf-6tisch-minimal] duty cycle mode. The permitted practice is to use a private, per-installation beacon authentication key.

8. Privacy Considerations

This specification relies on the uniqueness of EUI64 that is transferred in clear as part of the security context identifier. 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 EUI64 are minimal. In addition, the join response message contains an optional short address which can be assigned by JCE to JN. Short address is independent of the long-term identifier EUI64 and is encrypted in the response. For that reason, it is not possible to correlate the short address with the EUI64 used during the join. Use of short addresses once the join protocol completes mitigates the aforementioned privacy risks. In addition, EDHOC may be used for identity protection during the join protocol by generating a random context identifier in place of the EUI64 [I-D.selander-ace-cose-ecdhe].

9. IANA Considerations

There is no IANA action required for this document.

10. 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 the draft. The authors would also like to thank Francesca Palombini and Ludwig Seitz for participating in the discussions that have helped shape the document.

11. References

11.1. Normative References

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

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

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

[I-D.ietf-cose-msg] Schaad, J., "CBOR Object Signing and Encryption (COSE)", draft-ietf-cose-msg-24 (work in progress), November 2016.

[I-D.ietf-core-object-security] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-ietf-core-object-security-01 (work in progress), December 2016.

11.2. Informative References

[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, <http://www.rfc-editor.org/info/rfc7554>.

[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, <http://www.rfc-editor.org/info/rfc6775>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

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

[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, <http://www.rfc-editor.org/info/rfc4231>.

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

[I-D.ietf-6tisch-minimal] Vilajosana, X., Pister, K., and T. Watteyne, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-19 (work in progress), January 2017.

[I-D.ietf-6tisch-6top-protocol] Wang, Q. and X. Vilajosana, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-03 (work in progress), October 2016.

[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-08 (work in progress), December 2016.

[I-D.selander-ace-cose-ecdhe] Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-04 (work in progress), October 2016.

[IEEE802154-2015] IEEE standard for Information Technology, ., "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015.

Appendix A. Example

Figure 3 illustrates a join protocol exchange in case PSKs are used. JN instantiates the OSCOAP context and derives the traffic keys and nonces from the PSK. It uses the instantiated context to protect the CoAP request addressed with Proxy-Scheme option and well-known host name of JCE in the Uri-Host option. The example assumes a JA that is

already aware of JCE's IPv6 address and does not need to resolve the well-known "6tisch.jce" host name. Triggered by the presence of Proxy-Scheme option, JA forwards the request to the JCE. Once JCE receives the request, it looks up the correct context based on the context identifier (cid) field. It reconstructs OSCOAP's external Additional Authenticated Data (AAD) needed for verification based on:

- Version field of the received CoAP header.
- Code field of the received CoAP header.
- Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msg]
- Request URI reconstructed following [I-D.ietf-core-object-security].

Replay protection is ensured by OSCOAP and the tracking of sequence numbers at each side. In the example below, the response contains sequence number 7 meaning that there have already been some attempts to join under a given context, not coming from the JN. Once JA receives the response, it looks up and decodes the cid field in order to decide where to forward it. JA constructs the CoAP response to JN by setting the CoAP token to the value decoded from cid and constructs the link-local IPv6 address of JN from the EUI64 address found in the cid. Note that JA does not posses the key to decrypt the COSE object present in the payload so the join_response object is opaque to it. The response is matched to the request and verified for replay protection at JN using OSCOAP processing rules. Namely, to verify the response JN reconstructs the AAD based on:

- Version field of the received CoAP header.
- Code field of the received CoAP header.
- Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msg].
- Transaction identifier (Tid) of the corresponding CoAP request. Tid contains the context identifier (origin_info object), Sender ID (0x00 for JN), and Sender Sequence number (set to 1 in the example).

In addition to AAD, JN also uses the explicit, protected fields in the COSE message, present in the payload of the response. For more details, see [I-D.ietf-core-object-security] and [I-D.ietf-cose-msg].

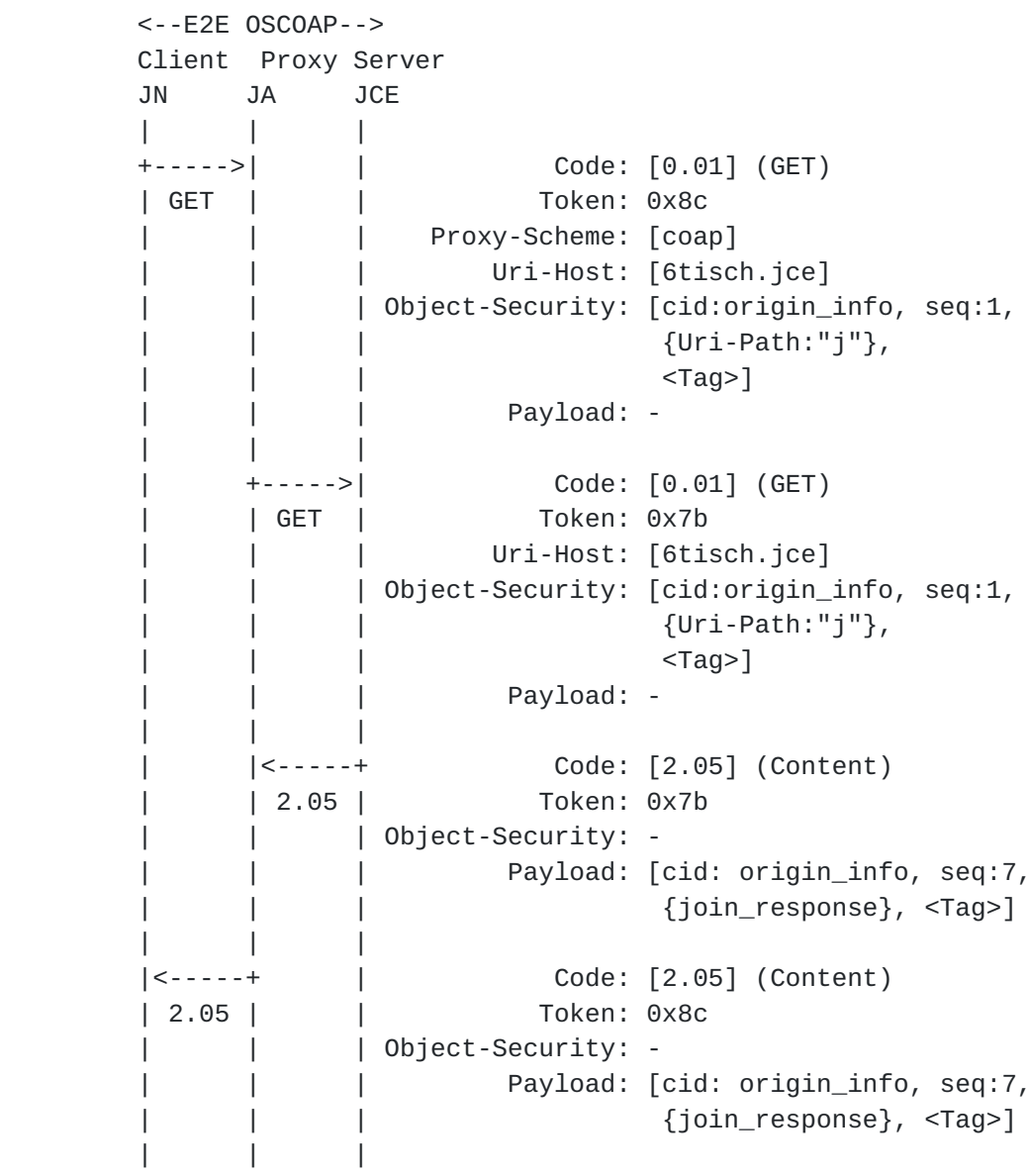

Figure 3: Example of a join protocol exchange with a PSK. {} denotes encryption and authentication, [] denotes authentication.

Where origin_info and join_response are as follows.

```
origin_info:
[
    h'00170d00060d9f0e', / JN's EUI64 /
    49152, / JN's UDP source port /
    0x8c   / JN's CoAP token /
]
```

Encodes to h'834800170d00060d9f0e19c000188c' with a size of 15 bytes.

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

Encodes to h'8281a201042050e6bf4287c2d7618d6a9687445ffd33e642af93' with a size of 26 bytes.

Authors' Addresses

Malisa Vucinic
Inria
2 Rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

Jonathan Simon
Linear Technology
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jsimon@linear.com

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

Email: pister@eecs.berkeley.edu