                   Secure group communication for CoAP
                   draft-ietf-core-oscore-groupcomm-02

Abstract

   This document describes a mode for protecting group communication
   over the Constrained Application Protocol (CoAP). The proposed mode
   relies on Object Security for Constrained RESTful Environments
   (OSCORE) and the CBOR Object Signing and Encryption (COSE) format.
   In particular, it is defined how OSCORE should be used in a group
   communication setting, while fulfilling the same security
   requirements for request messages and related response messages.
   Source authentication of all messages exchanged within the group is
   ensured, by means of digital signatures produced through private keys
   of sender endpoints and embedded in the protected CoAP messages.

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 https://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 December 30, 2018.

Table of Contents

1. Introduction

   The Constrained Application Protocol (CoAP) [RFC7252] is a web
   transfer protocol specifically designed for constrained devices and
   networks [RFC7228].

   Group communication for CoAP [RFC7390] addresses use cases where
   deployed devices benefit from a group communication model, for
   example to reduce latencies and improve performance. Use cases
   include lighting control, integrated building control, software and
   firmware updates, parameter and configuration updates, commissioning
   of constrained networks, and emergency multicast (see Appendix B).
   Furthermore, [RFC7390] recognizes the importance to introduce a
   secure mode for CoAP group communication. This specification defines
   such a mode.

   Object Security for Constrained RESTful Environments
   (OSCORE)[I-D.ietf-core-object-security] describes a security protocol
   based on the exchange of protected CoAP messages. OSCORE builds on
   CBOR Object Signing and Encryption (COSE) [RFC8152] and provides end-
   to-end encryption, integrity, and replay protection between a sending
   endpoint and a receiving endpoint possibly involving intermediary
   endpoints. To this end, a CoAP message is protected by including its
   payload (if any), certain options, and header fields in a COSE
   object, which finally replaces the authenticated and encrypted fields
   in the protected message.

   This document describes group OSCORE, providing end-to-end security
   of CoAP messages exchanged between members of a group. In
   particular, the described approach defines how OSCORE should be used
   in a group communication setting, so that end-to-end security is
   assured by using the same security method. That is, end-to-end
   security is assured for (multicast) CoAP requests sent by client
   endpoints to the group and for related CoAP responses sent as reply
   by multiple server endpoints. Group OSCORE provides source
   authentication of all CoAP messages exchanged within the group, by
   means of digital signatures produced through private keys of sender
   devices and embedded in the protected CoAP messages. As in OSCORE,
   it is still possible to simultaneously rely on DTLS to protect hop-
   by-hop communication between a sender endpoint and a proxy (and vice
   versa), and between a proxy and a recipient endpoint (and vice
   versa).

1.1. 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.

   Readers are expected to be familiar with the terms and concepts
   described in CoAP [RFC7252] including "endpoint", "client", "server",
   "sender" and "recipient"; group communication for CoAP [RFC7390];
   COSE and counter signatures [RFC8152].

   Readers are also expected to be familiar with the terms and concepts
   for protection and processing of CoAP messages through OSCORE, such
   as "Security Context" and "Master Secret", defined in
   [I-D.ietf-core-object-security].

   Terminology for constrained environments, such as "constrained
   device", "constrained-node network", is defined in [RFC7228].

   This document refers also to the following terminology.

   o Keying material: data that is necessary to establish and maintain
      secure communication among endpoints. This includes, for
      instance, keys and IVs [RFC4949].

   o Group: a set of endpoints that share group keying material and
      parameters (Common Context of the group's Security Context, see
      Section 2). That is, the term group used in this specification
      refers to a "security group", not to be confused with network/
      multicast groups or application groups.

   o Group Manager (GM): entity responsible for a set of OSCORE groups.
      Each endpoint in a group securely communicates with the respective
      GM, which is not required to be an actual group member and to take
      part in the group communication. The full list of
      responsibilities of the Group Manager is provided in Section 6.

   o Silent server: member of a group that never replies back after
      receiving request messages.

   o Group ID: group identifier assigned to the group. Group IDs are
      unique within the set of groups of a same Group Manager.

   o Endpoint ID: Sender ID of the endpoint, as defined in
      [I-D.ietf-core-object-security]. An Endpoint ID is provided to an
      endpoint upon joining a group, is valid only within that group,

      and is unique within the same group. Endpoints which are
      configured only as silent servers do not have an Endpoint ID.

   o Group request: CoAP request message sent by a client endpoint in
      the group to all server endpoints in that group.

   o Source authentication: evidence that a received message in the
      group originated from a specifically identified group member.
      This also provides assurances that the message was not tampered
      with by a different group member or by a non-group member.

2. OSCORE Security Context

   To support group communication secured with OSCORE, each endpoint
   registered as member of a group maintains a Security Context as
   defined in Section 3 of [I-D.ietf-core-object-security]. Each
   endpoint in a group stores:

   1. one Common Context, shared by all the endpoints in the group. In
       particular:

       * All the endpoints in the group agree on the same COSE AEAD
          algorithm.

       * The ID Context parameter stores the Group ID of the group,
          which is used to retrieve the Security Context for processing
          messages intended to the group's endpoints (see Section 4).
          The choice of the Group ID for a given group's Security
          Context is application specific. An example of specific
          formatting of the Group ID that would follow this
          specification is given in Appendix C. It is the role of the
          application to specify how to handle possible collisions.

       * A new parameter Counter Signature Algorithm is included, and
          its value identifies the algorithm used for source
          authenticating messages sent within the group, by means of a
          counter signature (see Section 4.5 of [RFC8152]). Its value
          is immutable once the Common Context is established. All the
          endpoints in the group agree on the same counter signature
          algorithm. The list of supported signature algorithms is part
          of the group communication policy and MUST include the EdDSA
          signature algorithm ed25519 [RFC8032].

   2. one Sender Context, unless the endpoint is configured exclusively
       as silent server. The Sender Context is used to secure outgoing
       group messages and is initialized according to Section 3 of
       [I-D.ietf-core-object-security], once the endpoint has joined the
       group. In practice, the symmetric keying material in the Sender

Context of the sender endpoint is shared with all the recipient
endpoints that have received group messages from that same sender
endpoint. Besides, in addition to what is defined in
[I-D.ietf-core-object-security], the Sender Context stores also
the endpoint's public-private key pair.

3. one Recipient Context for each distinct endpoint from which group
    messages are received, used to process such incoming messages.
    The recipient endpoint creates a new Recipient Context upon
    receiving an incoming message from another endpoint in the group
    for the first time (see Section 4.2 and Section 4.4). In
    practice, the symmetric keying material in a given Recipient
    Context of the recipient endpoint is shared with the associated
    sender endpoint from which group messages are received. Besides,
    in addition to what is defined in
    [I-D.ietf-core-object-security], each Recipient Context stores
    also the public key of the associated other endpoint from which
    group messages are received.

The table in Figure 1 overviews the new information included in the
OSCORE Security Context, with respect to what defined in Section 3 of
[I-D.ietf-core-object-security].

```
+--------------------------+----------------------------+
|      Context portion     |      New information       |
+--------------------------+----------------------------+
|                          |                            |
|      Common Context      | Counter signature algorithm|
|                          |                            |
|      Sender Context      | Endpoint's own private key |
|                          |                            |
|      Sender Context      | Endpoint's own public key  |
|                          |                            |
|   Each Recipient Context | Public key of the          |
|                          | associated other endpoint  |
|                          |                            |
+--------------------------+----------------------------+
```

Figure 1: Additions to the OSCORE Security Context

Upon receiving a secure CoAP message, a recipient endpoint relies on
the sender endpoint's public key, in order to verify the counter
signature conveyed in the COSE Object.

If not already stored in the Recipient Context associated to the
sender endpoint, the recipient endpoint retrieves the public key from
a trusted key repository. In such a case, the correct binding
between the sender endpoint and the retrieved public key must be

assured, for instance by means of public key certificates. Further
discussion about how public keys can be handled and retrieved in the
group is provided in Appendix D.2.

The Sender Key/IV stored in the Sender Context and the Recipient
Keys/IVs stored in the Recipient Contexts are derived according to
the same scheme defined in Section 3.2 of
[I-D.ietf-core-object-security].

2.1. Management of Group Keying Material

The approach described in this specification should take into account
the risk of compromise of group members. In particular, the adoption
of key management schemes for secure revocation and renewal of
Security Contexts and group keying material should be considered.

Consistently with the security assumptions in Appendix A.1, it is
RECOMMENDED to adopt a group key management scheme, and securely
distribute a new value for the Master Secret parameter of the group's
Security Context, before a new joining endpoint is added to the group
or after a currently present endpoint leaves the group. This is
necessary in order to preserve backward security and forward security
in the group.

In particular, a new Group Identifier (Gid) for that group and a new
value for the Master Secret parameter must also be distributed. An
example of Group Identifier format supporting this operation is
provided in Appendix C. Then, each group member re-derives the
keying material stored in its own Sender Context and Recipient
Contexts as described in Section 2, using the updated Group
Identifier.

Especially in dynamic, large-scale, groups where endpoints can join
and leave at any time, it is important that the considered group key
management scheme is efficient and highly scalable with the group
size, in order to limit the impact on performance due to the Security
Context and keying material update.

3. The COSE Object

When creating a protected CoAP message, an endpoint in the group
computes the COSE object using the untagged COSE_Encrypt0 structure
[RFC8152] as defined in Section 5 of [I-D.ietf-core-object-security],
with the following modifications.

o The value of the 'kid' parameter in the 'unprotected' field of
   response messagess SHALL be set to the Endpoint ID of the endpoint
   transmitting the message, i.e. the Sender ID.

   o The 'unprotected' field SHALL additionally include the following
      parameter:

      * CounterSignature0 : its value is set to the counter signature
         of the COSE object, computed by the endpoint by means of its
         own private key as described in Section 4.5 of [RFC8152]. The
         presence of this parameter is explicitly signaled, by using the
         reserved sixth least significant bit of the first byte of flag
         bits in the value of the OSCORE Option (see Section 6.1 of
         [I-D.ietf-core-object-security]).

   o The Additional Authenticated Data (AAD) considered to compute the
      COSE object is extended with the counter signature algorithm used
      to protect group messages. In particular, with reference to
      Section 5.4 of [I-D.ietf-core-object-security], the 'algorithms'
      array in the external_aad SHALL also include 'alg_countersign',
      which contains the Counter Signature Algorithm from the Common
      Context (see Section 2).

```
external_aad = [
   ...
   algorithms : [alg_aead : int / tstr , alg_countersign : int / tstr],
   ...
]
```

   o The OSCORE compression defined in Section 6 of
      [I-D.ietf-core-object-security] is used, with the following
      additions for the encoding of the OSCORE Option.

      * The fourth least significant bit of the first byte of flag bits
         SHALL be set to 1, to indicate the presence of the 'kid'
         parameter for both group requests and responses.

      * The fifth least significant bit of the first byte of flag bits
         MUST be set to 1 for group requests, to indicate the presence
         of the 'kid context' parameter in the OSCORE payload. The kid
         context flag MAY be set to 1 for responses.

      * The sixth least significant bit of the first byte of flag bits
         is originally marked as reserved in
         [I-D.ietf-core-object-security] and its usage is defined in
         this specification. This bit is set to 1 if the
         'CounterSignature0' parameter is present, or to 0 otherwise.
         In order to ensure source authentication of group messages as
         described in this specification, this bit SHALL be set to 1.

      * The 'kid context' value encodes the Group Identifier value
         (Gid) of the group's Security Context.

* The following q bytes (q given by the Counter Signature
  Algorithm specified in the Security Context) encode the value
  of the 'CounterSignature0' parameter including the counter
  signature of the COSE object.

* The remaining bytes in the OSCORE Option value encode the value
  of the 'kid' parameter, which is always present both in group
  requests and in responses.

```
 0 1 2 3 4 5 6 7 <----------- n bytes -----------> <-- 1 byte -->
+-+-+-+-+-+-+-+-+-------------------------------+--------------+
|0 0|1|h|1|  n  |       Partial IV (if any)     |  s (if any)  |
+-+-+-+-+-+-+-+-+-------------------------------+--------------+

 <------ s bytes ------> <--------- q bytes --------->
 ---------------------+---------------------------+-----------+
   kid context = Gid  |      CounterSignature0     |    kid    |
 ---------------------+---------------------------+-----------+
```
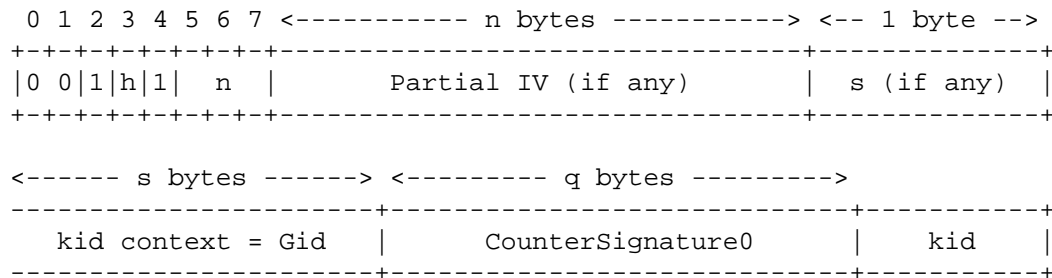
                    Figure 2: OSCORE Option Value

3.1. Example: Request

   Request with kid = 0x25, Partial IV = 5 and kid context = 0x44616c,
   assuming the label for the new kid context defined in
   [I-D.ietf-core-object-security] has value 10. COUNTERSIGN is the
   CounterSignature0 byte string as described in Section 3 and is 64
   bytes long in this example. The ciphertext in this example is 14
   bytes long.

   Before compression (96 bytes):

   [
   h'',
   { 4:h'25', 6:h'05', 10:h'44616c', 9:COUNTERSIGN },
   h'aea0155667924dff8a24e4cb35b9'
   ]

   After compression (85 bytes):

   Flag byte: 0b00111001 = 0x39

   Option Value: 39 05 03 44 61 6c COUNTERSIGN 25 (7 bytes + size of
    COUNTERSIGN)

   Payload: ae a0 15 56 67 92 4d ff 8a 24 e4 cb 35 b9 (14 bytes)

3.2. Example: Response

   Response with kid = 0x52. COUNTERSIGN is the CounterSignature0 byte
   string as described in Section 3 and is 64 bytes long in this
   example. The ciphertext in this example is 14 bytes long.

   Before compression (88 bytes):

   [
   h'',
   { 4:h'52', 9:COUNTERSIGN },
   h'60b035059d9ef5667c5a0710823b'
   ]

   After compression (80 bytes):

   Flag byte: 0b00101000 = 0x28

   Option Value: 28 COUNTERSIGN 52 (2 bytes + size of COUNTERSIGN)

   Payload: 60 b0 35 05 9d 9e f5 66 7c 5a 07 10 82 3b (14 bytes)

4. Message Processing

   Each request message and response message is protected and processed
   as specified in [I-D.ietf-core-object-security], with the
   modifications described in the following sections. The following
   security objectives are fulfilled, as further discussed in
   Appendix A.2: data replay protection, group-level data
   confidentiality, source authentication, message integrity, and
   message ordering.

   Furthermore, endpoints in the group locally perform error handling
   and processing of invalid messages according to the same principles
   adopted in [I-D.ietf-core-object-security]. However, a receiver
   endpoint MUST stop processing and silently reject any message which
   is malformed and does not follow the format specified in Section 3,
   without sending back any error message. This prevents servers from
   replying with multiple error messages to a client sending a group
   request, so avoiding the risk of flooding and possibly congesting the
   group.

4.1. Protecting the Request

   A client transmits a secure group request as described in Section 8.1
   of [I-D.ietf-core-object-security], with the following modifications.

o In step 2, the 'algorithms' array in the Additional Authenticated
   Data is modified as described in Section 3.

o In step 4, the encoding of the compressed COSE object is modified
   as described in Section 3.

4.2. Verifying the Request

   Upon receiving a secure group request, a server proceeds as described
   in Section 8.2 of [I-D.ietf-core-object-security], with the following
   modifications.

o In step 2, the decoding of the compressed COSE object is modified
   as described in Section 3. If the received Recipient ID ('kid')
   does not match with any Recipient Context for the retrieved Group
   ID ('kid context'), then the server creates a new Recipient
   Context, initializes it according to Section 3 of
   [I-D.ietf-core-object-security], and includes the client's public
   key.

o In step 4, the 'algorithms' array in the Additional Authenticated
   Data is modified as described in Section 3.

o In step 6, the server also verifies the counter signature using
   the public key of the client from the associated Recipient
   Context.

4.3. Protecting the Response

   A server that has received a secure group request may reply with a
   secure response, which is protected as described in Section 8.3 of
   [I-D.ietf-core-object-security], with the following modifications.

o In step 2, the 'algorithms' array in the Additional Authenticated
   Data is modified as described in Section 3.

o In step 4, the encoding of the compressed COSE object is modified
   as described in Section 3.

4.4. Verifying the Response

   Upon receiving a secure response message, the client proceeds as
   described in Section 8.4 of [I-D.ietf-core-object-security], with the
   following modifications.

o In step 2, the decoding of the compressed COSE object is modified
   as described in Section 3. If the received Recipient ID ('kid')
   does not match with any Recipient Context for the retrieved Group

ID ('kid context'), then the client creates a new Recipient
Context, initializes it according to Section 3 of
[I-D.ietf-core-object-security], and includes the server's public
key.

o In step 3, the 'algorithms' array in the Additional Authenticated
   Data is modified as described in Section 3.

o In step 5, the client also verifies the counter signature using
   the public key of the server from the associated Recipient
   Context.

5. Synchronization of Sequence Numbers

   Upon joining the group, new servers are not aware of the sequence
   number values currently used by different clients to transmit group
   requests. This means that, when such servers receive a secure group
   request from a given client for the first time, they are not able to
   verify if that request is fresh and has not been replayed. The same
   holds when a server loses synchronization with sequence numbers of
   clients, for instance after a device reboot.

   The exact way to address this issue depends on the specific use case
   and its synchronization requirements. The list of methods to handle
   synchronization of sequence numbers is part of the group
   communication policy, and different servers can use different
   methods. Appendix E describes three possible approaches that can be
   considered.

6. Responsibilities of the Group Manager

   The Group Manager is responsible for performing the following tasks:

   o Creating and managing OSCORE groups. This includes the assignment
      of a Group ID to every newly created group, as well as ensuring
      uniqueness of Group IDs within the set of its OSCORE groups.

   o Defining policies for authorizing the joining of its OSCORE
      groups. Such policies can be enforced by a third party, which is
      in a trust relation with the Group Manager and enforces join
      policies on behalf of the Group Manager.

   o Driving the join process to add new endpoints as group members.

   o Establishing Security Common Contexts and providing them to
      authorized group members during the join process, together with a
      corresponding Security Sender Context.

o Generating and managing Endpoint IDs within its OSCORE groups, as
   well as assigning and providing them to new endpoints during the
   join process. This includes ensuring uniqueness of Endpoints IDs
   within each of its OSCORE groups.

o Defining a set of supported signature algorithms as part of the
   communication policy of each of its OSCORE groups, and signalling
   it to new endpoints during the join process.

o Defining the methods to handle loss of synchronization with
   sequence numbers as part of the communication policy of each of
   its OSCORE groups, and signaling the one(s) to use to new
   endpoints during the join process.

o Renewing the Security Context of an OSCORE group upon membership
   change, by revoking and renewing common security parameters and
   keying material (rekeying).

o Providing the management keying material that a new endpoint
   requires to participate in the rekeying process, consistently with
   the key management scheme used in the group joined by the new
   endpoint.

o Updating the Group ID of its OSCORE groups, upon renewing the
   respective Security Context.

The Group Manager may additionally be responsible for the following
tasks:

o Acting as trusted key repository, in order to store the public
   keys of the members of its OSCORE groups, and provide such public
   keys to other members of the same group upon request. This
   specification recommends that the Group Manager is entrusted to
   perform this task.

o Acting as network router device where endpoints register to
   correctly receive group messages sent to the multicast IP address
   of that group.

o Autonomously and locally enforcing access policies to authorize
   new endpoints to join its OSCORE groups.

7. Security Considerations

The same security considerations from OSCORE (Section 11 of
[I-D.ietf-core-object-security]) apply to this specification.
Additional security aspects to be taken into account are discussed
below.

7.1. Group-level Security

   The approach described in this document relies on commonly shared
   group keying material to protect communication within a group. This
   means that messages are encrypted at a group level (group-level data
   confidentiality), i.e. they can be decrypted by any member of the
   group, but not by an external adversary or other external entities.

   In addition, it is required that all group members are trusted, i.e.
   they do not forward the content of group messages to unauthorized
   entities. However, in many use cases, the devices in the group
   belong to a common authority and are configured by a commissioner
   (see Appendix B).

7.2. Uniqueness of (key, nonce)

   The proof for uniqueness of (key, nonce) pairs in Appendix D.3 of
   [I-D.ietf-core-object-security] is also valid in group communication
   scenarios. That is, given an OSCORE group:

   o Uniqueness of Sender IDs within the group is enforced by the Group
      Manager.

   o Case A is limited to requests, and same considerations hold.

   o Case B applies to all responses, and same considerations hold.

   It follows that each message encrypted/decrypted with the same Sender
   Key is processed by using a different (ID_PIV, PIV) pair. This means
   that nonces used by any fixed encrypting endpoint are unique. Thus,
   each message is processed with a different (key, nonce) pair.

7.3. Collision of Group Identifiers

   In case endpoints are deployed in multiple groups managed by
   different non-synchronized Group Managers, it is possible for Group
   Identifiers of different groups to coincide. However, this does not
   impair the security of the AEAD algorithm.

   In fact, as long as the Master Secret is different for different
   groups and this condition holds over time, and as long as the Sender
   IDs within a group are unique, it follows that AEAD keys and nonces
   are different among different groups.

8. IANA Considerations

   This document has no actions for IANA.

9. Acknowledgments

   The authors sincerely thank Stefan Beck, Rolf Blom, Carsten Bormann,
   Esko Dijk, Klaus Hartke, Richard Kelsey, John Mattsson, Jim Schaad,
   Ludwig Seitz and Peter van der Stok for their feedback and comments.

   The work on this document has been partly supported by the EIT-
   Digital High Impact Initiative ACTIVE.

10. References

10.1. Normative References

   [I-D.ietf-core-object-security]
              Selander, G., Mattsson, J., Palombini, F., and L. Seitz,
              "Object Security for Constrained RESTful Environments
              (OSCORE)", draft-ietf-core-object-security-13 (work in
              progress), June 2018.

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

   [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>.

   [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital
              Signature Algorithm (EdDSA)", RFC 8032,
              DOI 10.17487/RFC8032, January 2017,
              <https://www.rfc-editor.org/info/rfc8032>.

   [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>.

10.2. Informative References

   [I-D.ietf-ace-dtls-authorize]
             Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and
             L. Seitz, "Datagram Transport Layer Security (DTLS)
             Profile for Authentication and Authorization for
             Constrained Environments (ACE)", draft-ietf-ace-dtls-
             authorize-03 (work in progress), March 2018.

   [I-D.ietf-ace-oauth-authz]
             Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and
             H. Tschofenig, "Authentication and Authorization for
             Constrained Environments (ACE) using the OAuth 2.0
             Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-12
             (work in progress), May 2018.

   [I-D.ietf-ace-oscore-profile]
             Seitz, L., Palombini, F., Gunnarsson, M., and G. Selander,
             "OSCORE profile of the Authentication and Authorization
             for Constrained Environments Framework", draft-ietf-ace-
             oscore-profile-01 (work in progress), March 2018.

   [I-D.ietf-core-echo-request-tag]
             Amsuess, C., Mattsson, J., and G. Selander, "Echo and
             Request-Tag", draft-ietf-core-echo-request-tag-01 (work in
             progress), March 2018.

   [I-D.palombini-ace-key-groupcomm]
             Palombini, F. and M. Tiloca, "Key Provisioning for Group
             Communication using ACE", draft-palombini-ace-key-
             groupcomm-01 (work in progress), June 2018.

   [I-D.somaraju-ace-multicast]
             Somaraju, A., Kumar, S., Tschofenig, H., and W. Werner,
             "Security for Low-Latency Group Communication", draft-
             somaraju-ace-multicast-02 (work in progress), October
             2016.

   [I-D.tiloca-ace-oscoap-joining]
             Tiloca, M. and J. Park, "Joining OSCORE groups in ACE",
             draft-tiloca-ace-oscoap-joining-03 (work in progress),
             March 2018.

   [RFC2093] Harney, H. and C. Muckenhirn, "Group Key Management
             Protocol (GKMP) Specification", RFC 2093,
             DOI 10.17487/RFC2093, July 1997,
             <https://www.rfc-editor.org/info/rfc2093>.

   [RFC2094] Harney, H. and C. Muckenhirn, "Group Key Management
              Protocol (GKMP) Architecture", RFC 2094,
              DOI 10.17487/RFC2094, July 1997,
              <https://www.rfc-editor.org/info/rfc2094>.

   [RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for
              Multicast: Issues and Architectures", RFC 2627,
              DOI 10.17487/RFC2627, June 1999,
              <https://www.rfc-editor.org/info/rfc2627>.

   [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A.
              Thyagarajan, "Internet Group Management Protocol, Version
              3", RFC 3376, DOI 10.17487/RFC3376, October 2002,
              <https://www.rfc-editor.org/info/rfc3376>.

   [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security
              Architecture", RFC 3740, DOI 10.17487/RFC3740, March 2004,
              <https://www.rfc-editor.org/info/rfc3740>.

   [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener
              Discovery Version 2 (MLDv2) for IPv6", RFC 3810,
              DOI 10.17487/RFC3810, June 2004,
              <https://www.rfc-editor.org/info/rfc3810>.

   [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm,
              "Multicast Security (MSEC) Group Key Management
              Architecture", RFC 4046, DOI 10.17487/RFC4046, April 2005,
              <https://www.rfc-editor.org/info/rfc4046>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <https://www.rfc-editor.org/info/rfc4301>.

   [RFC4535] Harney, H., Meth, U., Colegrove, A., and G. Gross,
              "GSAKMP: Group Secure Association Key Management
              Protocol", RFC 4535, DOI 10.17487/RFC4535, June 2006,
              <https://www.rfc-editor.org/info/rfc4535>.

   [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>.

   [RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
              FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
              <https://www.rfc-editor.org/info/rfc4949>.

   [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6
              Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,
              DOI 10.17487/RFC6282, September 2011,
              <https://www.rfc-editor.org/info/rfc6282>.

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

   [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
              RFC 6749, DOI 10.17487/RFC6749, October 2012,
              <https://www.rfc-editor.org/info/rfc6749>.

   [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for
              Constrained-Node Networks", RFC 7228,
              DOI 10.17487/RFC7228, May 2014,
              <https://www.rfc-editor.org/info/rfc7228>.

   [RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for
              the Constrained Application Protocol (CoAP)", RFC 7390,
              DOI 10.17487/RFC7390, October 2014,
              <https://www.rfc-editor.org/info/rfc7390>.

Appendix A. Assumptions and Security Objectives

   This section presents a set of assumptions and security objectives
   for the approach described in this document.

A.1. Assumptions

   The following assumptions are assumed to be already addressed and are
   out of the scope of this document.

   o Multicast communication topology: this document considers both
      1-to-N (one sender and multiple recipients) and M-to-N (multiple
      senders and multiple recipients) communication topologies. The
      1-to-N communication topology is the simplest group communication
      scenario that would serve the needs of a typical low-power and
      lossy network (LLN). Examples of use cases that benefit from
      secure group communication are provided in Appendix B.

      In a 1-to-N communication model, only a single client transmits
      data to the group, in the form of request messages; in an M-to-N
      communication model (where M and N do not necessarily have the
      same value), M group members are clients. According to [RFC7390],
      any possible proxy entity is supposed to know about the clients in
      the group and to not perform aggregation of response messages from
      multiple servers. Also, every client expects and is able to

handle multiple response messages associated to a same request
sent to the group.

o Group size: security solutions for group communication should be
   able to adequately support different and possibly large groups.
   The group size is the current number of members in a group. In
   the use cases mentioned in this document, the number of clients
   (normally the controlling devices) is expected to be much smaller
   than the number of servers (i.e. the controlled devices). A
   security solution for group communication that supports 1 to 50
   clients would be able to properly cover the group sizes required
   for most use cases that are relevant for this document. The
   maximum group size is expected to be in the range of 2 to 100
   devices. Groups larger than that should be divided into smaller
   independent groups, e.g. by grouping lights in a building on a per
   floor basis.

o Communication with the Group Manager: an endpoint must use a
   secure dedicated channel when communicating with the Group
   Manager, even when not registered as group member. In particular,
   communications with the Group Manager occuring during the join
   process to become a group member must also be secured.

o Establishment and management of Security Contexts: an OSCORE
   Security Context must be established among the group members. In
   particular, a Common Context must be provided to a new joining
   endpoint together with a corresponding Sender Context. On the
   other hand, Recipient Contexts are locally and individually
   derived by each group member. A secure mechanism must be used to
   generate, revoke and (re-)distribute keying material, multicast
   security policies and security parameters in the group. The
   actual establishment and management of the Security Context is out
   of the scope of this document, and it is anticipated that an
   activity in IETF dedicated to the design of a generic key
   management scheme will include this feature, preferably based on
   [RFC3740][RFC4046][RFC4535].

o Multicast data security ciphersuite: all group members must agree
   on a ciphersuite to provide authenticity, integrity and
   confidentiality of messages in the group. The ciphersuite is
   specified as part of the Security Context.

o Backward security: a new device joining the group should not have
   access to any old Security Contexts used before its joining. This
   ensures that a new group member is not able to decrypt
   confidential data sent before it has joined the group. The
   adopted key management scheme should ensure that the Security
   Context is updated to ensure backward confidentiality. The actual

      mechanism to update the Security Context and renew the group
      keying material upon a group member's joining has to be defined as
      part of the group key management scheme.

   o Forward security: entities that leave the group should not have
      access to any future Security Contexts or message exchanged within
      the group after their leaving. This ensures that a former group
      member is not able to decrypt confidential data sent within the
      group anymore. Also, it ensures that a former member is not able
      to send encrypted and/or integrity protected messages to the group
      anymore. The actual mechanism to update the Security Context and
      renew the group keying material upon a group member's leaving has
      to be defined as part of the group key management scheme.

A.2. Security Objectives

   The approach described in this document aims at fulfilling the
   following security objectives:

   o Data replay protection: replayed group request messages or
      response messages must be detected.

   o Group-level data confidentiality: messages sent within the group
      shall be encrypted if privacy sensitive data is exchanged within
      the group. This document considers group-level data
      confidentiality since messages are encrypted at a group level,
      i.e. in such a way that they can be decrypted by any member of the
      group, but not by an external adversary or other external
      entities.

   o Source authentication: messages sent within the group shall be
      authenticated. That is, it is essential to ensure that a message
      is originated by a member of the group in the first place, and in
      particular by a specific member of the group.

   o Message integrity: messages sent within the group shall be
      integrity protected. That is, it is essential to ensure that a
      message has not been tampered with by an external adversary or
      other external entities which are not group members.

   o Message ordering: it must be possible to determine the ordering of
      messages coming from a single sender endpoint. In accordance with
      OSCORE [I-D.ietf-core-object-security], this results in providing
      relative freshness of group requests and absolute freshness of
      responses. It is not required to determine ordering of messages
      from different sender endpoints.

Appendix B. List of Use Cases

   Group Communication for CoAP [RFC7390] provides the necessary
   background for multicast-based CoAP communication, with particular
   reference to low-power and lossy networks (LLNs) and resource
   constrained environments. The interested reader is encouraged to
   first read [RFC7390] to understand the non-security related details.
   This section discusses a number of use cases that benefit from secure
   group communication. Specific security requirements for these use
   cases are discussed in Appendix A.

   o Lighting control: consider a building equipped with IP-connected
      lighting devices, switches, and border routers. The devices are
      organized into groups according to their physical location in the
      building. For instance, lighting devices and switches in a room
      or corridor can be configured as members of a single group.
      Switches are then used to control the lighting devices by sending
      on/off/dimming commands to all lighting devices in a group, while
      border routers connected to an IP network backbone (which is also
      multicast-enabled) can be used to interconnect routers in the
      building. Consequently, this would also enable logical groups to
      be formed even if devices in the lighting group may be physically
      in different subnets (e.g. on wired and wireless networks).
      Connectivity between lighting devices may be realized, for
      instance, by means of IPv6 and (border) routers supporting 6LoWPAN
      [RFC4944][RFC6282]. Group communication enables synchronous
      operation of a group of connected lights, ensuring that the light
      preset (e.g. dimming level or color) of a large group of
      luminaires are changed at the same perceived time. This is
      especially useful for providing a visual synchronicity of light
      effects to the user. As a practical guideline, events within a
      200 ms interval are perceived as simultaneous by humans, which is
      necessary to ensure in many setups. Devices may reply back to the
      switches that issue on/off/dimming commands, in order to report
      about the execution of the requested operation (e.g. OK, failure,
      error) and their current operational status. In a typical
      lighting control scenario, a single switch is the only entity
      responsible for sending commands to a group of lighting devices.
      In more advanced lighting control use cases, a M-to-N
      communication topology would be required, for instance in case
      multiple sensors (presence or day-light) are responsible to
      trigger events to a group of lighting devices. Especially in
      professional lighting scenarios, the roles of client and server
      are configured by the lighting commissioner, and devices strictly
      follow those roles.

   o Integrated building control: enabling Building Automation and
      Control Systems (BACSs) to control multiple heating, ventilation

and air-conditioning units to pre-defined presets. Controlled units can be organized into groups in order to reflect their physical position in the building, e.g. devices in the same room can be configured as members of a single group. As a practical guideline, events within intervals of seconds are typically acceptable. Controlled units are expected to possibly reply back to the BACS issuing control commands, in order to report about the execution of the requested operation (e.g. OK, failure, error) and their current operational status.

o Software and firmware updates: software and firmware updates often comprise quite a large amount of data. This can overload a LLN that is otherwise typically used to deal with only small amounts of data, on an infrequent base. Rather than sending software and firmware updates as unicast messages to each individual device, multicasting such updated data to a larger group of devices at once displays a number of benefits. For instance, it can significantly reduce the network load and decrease the overall time latency for propagating this data to all devices. Even if the complete whole update process itself is secured, securing the individual messages is important, in case updates consist of relatively large amounts of data. In fact, checking individual received data piecemeal for tampering avoids that devices store large amounts of partially corrupted data and that they detect tampering hereof only after all data has been received. Devices receiving software and firmware updates are expected to possibly reply back, in order to provide a feedback about the execution of the update operation (e.g. OK, failure, error) and their current operational status.

o Parameter and configuration update: by means of multicast communication, it is possible to update the settings of a group of similar devices, both simultaneously and efficiently. Possible parameters are related, for instance, to network load management or network access controls. Devices receiving parameter and configuration updates are expected to possibly reply back, to provide a feedback about the execution of the update operation (e.g. OK, failure, error) and their current operational status.

o Commissioning of LLNs systems: a commissioning device is responsible for querying all devices in the local network or a selected subset of them, in order to discover their presence, and be aware of their capabilities, default configuration, and operating conditions. Queried devices displaying similarities in their capabilities and features, or sharing a common physical location can be configured as members of a single group. Queried devices are expected to reply back to the commissioning device, in

order to notify their presence, and provide the requested
information and their current operational status.

o Emergency multicast: a particular emergency related information
   (e.g. natural disaster) is generated and multicast by an emergency
   notifier, and relayed to multiple devices. The latters may reply
   back to the emergency notifier, in order to provide their feedback
   and local information related to the ongoing emergency. This kind
   of setups should additionally rely on a fault tolerance multicast
   algorithm, such as MPL.

Appendix C. Example of Group Identifier Format

   This section provides an example of how the Group Identifier (Gid)
   can be specifically formatted. That is, the Gid can be composed of
   two parts, namely a Group Prefix and a Group Epoch.

   The Group Prefix is constant over time and is uniquely defined in the
   set of all the groups associated to the same Group Manager. The
   choice of the Group Prefix for a given group's Security Context is
   application specific. The size of the Group Prefix directly impact
   on the maximum number of distinct groups under the same Group
   Manager.

   The Group Epoch is set to 0 upon the group's initialization, and is
   incremented by 1 upon completing each renewal of the Security Context
   and keying material in the group (see Section 2.1). In particular,
   once a new Master Secret has been distributed to the group, all the
   group members increment by 1 the Group Epoch in the Group Identifier
   of that group.

   As an example, a 3-byte Group Identifier can be composed of: i) a
   1-byte Group Prefix '0xb1' interpreted as a raw byte string; and ii)
   a 2-byte Group Epoch interpreted as an unsigned integer ranging from
   0 to 65535. Then, after having established the Security Common
   Context 61532 times in the group, its Group Identifier will assume
   value '0xb1f05c'.

   As discussed in Section 7.3, if endpoints are deployed in multiple
   groups managed by different non-synchronized Group Managers, it is
   possible that Group Identifiers of different groups coincide at some
   point in time. In this case, a recipient endpoint has to handle
   coinciding Group Identifiers, and has to try using different OSCORE
   Security Contexts to process an incoming message, until the right one
   is found and the message is correctly verified. Therefore, it is
   favourable that Group Idenfiers from different Group Managers have a
   size that result in a small probability of collision. How small this
   probability should be is up to system designers.

Appendix D. Set-up of New Endpoints

   An endpoint joins a group by explicitly interacting with the
   responsible Group Manager. Communications between a joining endpoint
   and the Group Manager rely on the CoAP protocol and must be secured.
   Specific details on how to secure communications between joining
   endpoints and a Group Manager are out of scope.

   In order to receive multicast messages sent to the group, a joining
   endpoint has to register with a network router device
   [RFC3376][RFC3810], signaling its intent to receive packets sent to
   the multicast IP address of that group. As a particular case, the
   Group Manager can also act as such a network router device. Upon
   joining the group, endpoints are not required to know how many and
   what endpoints are active in the same group.

   Furthermore, in order to participate in the secure group
   communication, an endpoint needs to be properly initialized upon
   joining the group. In particular, the Group Manager provides keying
   material and parameters to a joining endpoint, which can then
   initialize its own Security Context (see Section 2).

   The following Appendix D.1 provides an example describing how such
   information can be provided to an endpoint upon joining a group
   through the responsible Group Manager. Then, Appendix D.2 discusses
   how public keys of group members can be handled and made available to
   group members. Finally, Appendix D.3 overviews how the ACE framework
   for Authentication and Authorization in constrained environments
   [I-D.ietf-ace-oauth-authz] can be possibly used to support such a
   join process.

D.1. Join Process

   An endpoint requests to join a group by sending a confirmable CoAP
   POST request to the Group Manager responsible for that group. This
   join request can reflect the format of the Key Distribution Request
   message defined in Section 4.1 of [I-D.palombini-ace-key-groupcomm].
   Besides, it can be addressed to a CoAP resource associated to that
   group and carries the following information.

   o Group identifier: the Group Identifier (Gid) of the group, as
      known to the joining endpoint at this point in time. This may not
      fully coincide with the Gid currently associated to the group,
      e.g. if it includes a dynamic component. This information can be
      mapped to the first element of the 'scope' parameter of the Key
      Distribution Request message defined in Section 4.1 of
      [I-D.palombini-ace-key-groupcomm].

   o Role: the exact role of the joining endpoint in the group.
      Possible values are: "client", "server", "silent server", "client
      and server", or "client and silent server". This information can
      be mapped to the second element of the 'scope' parameter of the
      Key Distribution Request message defined in Section 4.1 of
      [I-D.palombini-ace-key-groupcomm].

   o Retrieval flag: indication of interest to receive the public keys
      of the endpoints currently in the group, as included in the
      following join response. This flag must not be present if the
      Group Manager is not configured to store the public keys of group
      members, or if the joining endpoint is configured exclusively as
      silent server for the group to join. This information can be
      mapped to the 'get_pub_keys' parameter of the Key Distribution
      Request message defined in Section 4.1 of
      [I-D.palombini-ace-key-groupcomm].

   o Identity credentials: information elements to enforce source
      authentication of group messages from the joining endpoint, such
      as its public key. The exact content depends on whether the Group
      Manager is configured to store the public keys of group members.
      If this is the case, this information is omitted if it has been
      provided to the same Group Manager upon previously joining the
      same or a different group under its control. This information is
      also omitted if the joining endpoint is configured exclusively as
      silent server for the joined group. Appendix D.2 discusses
      additional details on provisioning of public keys and other
      information to enforce source authentication of joining
      endpoints's messages. This information can be mapped to the
      'client_cred' parameter of the Key Distribution Request message
      defined in Section 4.1 of [I-D.palombini-ace-key-groupcomm].

   The Group Manager must be able to verify that the joining endpoint is
   authorized to become a member of the group. To this end, the Group
   Manager can directly authorize the joining endpoint, or expect it to
   provide authorization evidence previously obtained from a trusted
   entity. Appendix D.3 describes how this can be achieved by
   leveraging the ACE framework for Authentication and Authorization in
   constrained environments [I-D.ietf-ace-oauth-authz].

   In case of successful authorization check, the Group Manager
   generates an Endpoint ID assigned to the joining endpoint, before
   proceeding with the rest of the join process. Instead, in case the
   authorization check fails, the Group Manager aborts the join process.
   Further details about the authorization of joining endpoint are out
   of scope.

As discussed in Section 2.1, it is recommended that the Security
Context is renewed before the joining endpoint receives the group
keying material and becomes a new active member of the group. This
is achieved by securely distributing a new Master Secret and a new
Group Identifier to the endpoints currently present in the same
group.

Once renewed the Security Context in the group, the Group Manager
replies to the joining endpoint with a CoAP response carrying the
following information. This join response can reflect the format of
the Key Distribution Response message defined in Section 4.2 of
[I-D.palombini-ace-key-groupcomm].

o Security Common Context: the OSCORE Security Common Context
   associated to the joined group (see Section 2). This information
   can be mapped to the 'key' parameter of the Key Distribution
   Response message defined in Section 4.2 of
   [I-D.palombini-ace-key-groupcomm].

o Endpoint ID: the Endpoint ID associated to the joining endpoint.
   This information is not included in case 'Role' in the join
   request is equal to "silent server". This information can be
   mapped to the 'clientID' parameter within the 'key' parameter of
   the Key Distribution Response message defined in Section 4.2 of
   [I-D.palombini-ace-key-groupcomm].

o Member public keys: the public keys of the endpoints currently
   present in the group. This includes: the public keys of the non-
   silent servers currently in the group, if the joining endpoint is
   configured (also) as client; and the public keys of the clients
   currently in the group, if the joining endpoint is configured
   (also) as server or silent server. This information is omitted in
   case the Group Manager is not configured to store the public keys
   of group members or if the 'Retrieval flag' was not present in the
   join request. Appendix D.2 discusses additional details on
   provisioning public keys upon joining the group and on retrieving
   public keys of group members. This information can be mapped to
   the 'pub_keys' parameter of the Key Distribution Response message
   defined in Section 4.2 of [I-D.palombini-ace-key-groupcomm].

o Group policies: a list of key words indicating the particular
   policies enforced in the group. This includes, for instance, the
   method to achieve synchronization of sequence numbers among group
   members (see Appendix E), as well as the rekeying protocol used to
   renew the keying material in the group (see Section 2.1). This
   information can be mapped to the 'group_policies' parameter of the
   Key Distribution Response message defined in Section 4.2 of
   [I-D.palombini-ace-key-groupcomm].

o Management keying material: the set of administrative keying
   material used to participate in the group rekeying process run by
   the Group Manager (see Section 2.1). The specific elements of
   this management keying material depend on the group rekeying
   protocol used in the group. For instance, this can simply consist
   in a group key encryption key and a pairwise symmetric key shared
   between the joining endpoint and the Group Manager, in case GKMP
   [RFC2093][RFC2094] is used. Instead, if key-tree based rekeying
   protocols like LKH [RFC2627] are used, it can consist in the set
   of symmetric keys associated to the key-tree leaf representing the
   group member up to the key-tree root representing the group key
   encryption key. This information can be mapped to the
   'mgt_key_material' parameter of the Key Distribution Response
   message defined in Section 4.2 of
   [I-D.palombini-ace-key-groupcomm].

D.2. Provisioning and Retrieval of Public Keys

   As mentioned in Section 6, it is recommended that the Group Manager
   acts as trusted key repository, so storing public keys of group
   members and providing them to other members of the same group upon
   request. In such a case, a joining endpoint provides its own public
   key to the Group Manager, as 'Identity credentials' of the join
   request, when joining the group (see Appendix D.1).

   After that, the Group Manager should verify that the joining endpoint
   actually owns the associated private key, for instance by performing
   a proof-of-possession challenge-response, whose details are out of
   scope. In case of failure, the Group Manager performs up to a pre-
   defined maximum number of retries, after which it aborts the join
   process.

   In case of successful challenge-response, the Group Manager stores
   the received public key as associated to the joining endpoint and its
   Endpoint ID. From then on, that public key will be available for
   secure and trusted delivery to other endpoints in the group. A
   possible approach for a group member to retrieve the public key of
   other group members is described in Section 7 of
   [I-D.palombini-ace-key-groupcomm].

   Finally, the Group Manager sends the join response to the joining
   endpoint, as described in Appendix D.1.

   The joining endpoint does not have to provide its own public key if
   that already occurred upon previously joining the same or a different
   group under the same Group Manager. However, separately for each
   group under its control, the Group Manager maintains an updated list

of active Endpoint IDs associated to the respective endpoint's public key.

Instead, in case the Group Manager does not act as trusted key repository, the following exchange with the Group Manager can occur during the join process.

1. The joining endpoint signs its own certificate by using its own private key. The certificate includes also the identifier of the issuer Certification Authority (CA). There is no restriction on the Certificate Subject included in the joining endpoint's certificate.

2. The joining endpoint specifies the signed certificate as 'Identity credentials' in the join request (Appendix D.1). The joining endpoint can optionally specify also a list of public key repositories storing its own certificate. In such a case, this information can be mapped to the 'pub_keys_repos' parameter of the Key Distribution Request message defined in Section 4.1 of [I-D.palombini-ace-key-groupcomm].

3. When processing the join request, the Group Manager first validates the certificate by verifying the signature of the issuer CA, and then verifies the signature of the joining endpoint.

4. The Group Manager stores the association between the Certificate Subject of the joining endpoint's certificate and the pair {Group ID, Endpoint ID of the joining endpoint}. If received from the joining endpoint, the Group Manager also stores the list of public key repositories storing the certificate of the joining endpoint.

When a group member X wants to retrieve the public key of another group member Y in the same group, the endpoint X proceeds as follows.

1. The endpoint X contacts the Group Manager, specifying the pair {Group ID, Endpoint ID of the endpoint Y}.

2. The Group Manager provides the endpoint X with the Certificate Subject CS from the certificate of endpoint Y. If available, the Group Manager provides the endpoint X also with the list of public key repositories storing the certificate of the endpoint Y.

3. The endpoint X retrieves the certificate of the endpoint X from a key repository storing it, by using the Certificate Subject CS.

D.3. Group Joining Based on the ACE Framework

   The join process to register an endpoint as a new member of a group
   can be based on the ACE framework for Authentication and
   Authorization in constrained environments [I-D.ietf-ace-oauth-authz],
   built on re-use of OAuth 2.0 [RFC6749].

   In particular, the approach described in
   [I-D.tiloca-ace-oscoap-joining] uses the ACE framework to delegate
   the authentication and authorization of joining endpoints to an
   Authorization Server in a trust relation with the Group Manager. At
   the same time, it allows a joining endpoint to establish a secure
   channel with the Group Manager, by leveraging protocol-specific
   profiles of ACE, such as [I-D.ietf-ace-oscore-profile] and
   [I-D.ietf-ace-dtls-authorize], to achieve communication security,
   proof-of-possession and server authentication.

   More specifically and with reference to the terminology defined in
   OAuth 2.0:

   o The joining endpoint acts as ACE Client;

   o The Group Manager acts as ACE Resource Server, with different CoAP
      resources for different groups it is responsible for;

   o An Authorization Server enables and enforces authorized access of
      the joining endpoint to the Group Manager and its CoAP resources
      paired with groups to join.

   Messages exchanged among the participants follow the formats defined
   in [I-D.palombini-ace-key-groupcomm]. Both the joining endpoint and
   the Group Manager have to adopt secure communication also for any
   message exchange with the Authorization Server. To this end,
   different alternatives are possible, such as OSCORE, DTLS [RFC6347]
   or IPsec [RFC4301].

Appendix E. Examples of Synchronization Approaches

   This section describes three possible approaches that can be
   considered by server endpoints to synchronize with sequence numbers
   of client endpoints sending group requests.

E.1. Best-Effort Synchronization

   Upon receiving a group request from a client, a server does not take
   any action to synchonize with the sequence number of that client.
   This provides no assurance at all as to message freshness, which can
   be acceptable in non-critical use cases.

E.2. Baseline Synchronization

   Upon receiving a group request from a given client for the first
   time, a server initializes its last-seen sequence number in its
   Recipient Context associated to that client. However, the server
   drops the group request without delivering it to the application
   layer. This provides a reference point to identify if future group
   requests from the same client are fresher than the last one received.

   A replay time interval exists, between when a possibly replayed
   message is originally transmitted by a given client and the first
   authentic fresh message from that same client is received. This can
   be acceptable for use cases where servers admit such a trade-off
   between performance and assurance of message freshness.

E.3. Challenge-Response Synchronization

   A server performs a challenge-response exchange with a client, by
   using the Echo Option for CoAP described in Section 2 of
   [I-D.ietf-core-echo-request-tag] and consistently with what specified
   in Section 7.5.2 of [I-D.ietf-core-object-security].

   That is, upon receiving a group request from a particular client for
   the first time, the server processes the message as described in
   Section 4.2 of this specification, but, even if valid, does not
   deliver it to the application. Instead, the server replies to the
   client with a 4.03 Forbidden response message including an Echo
   Option, and stores the option value included therein.

   Upon receiving a 4.03 Forbidden response that includes an Echo Option
   and originates from a verified group member, a client sends a request
   as a unicast message addressed to the same server, echoing the Echo
   Option value. In particular, the client does not necessarily resend
   the same group request, but can instead send a more recent one, if
   the application permits it. This makes it possible for the client to
   not retain previously sent group requests for full retransmission,
   unless the application explicitly requires otherwise. In either
   case, the client uses the sequence number value currently stored in
   its own Sender Context. If the client stores group requests for
   possible retransmission with the Echo Option, it should not store a
   given request for longer than a pre-configured time interval. Note
   that the unicast request echoing the Echo Option is correctly treated
   and processed as a group message, since the 'kid context' field
   including the Group Identifier of the OSCORE group is still present
   in the OSCORE Option as part of the COSE object (see Section 3).

   Upon receiving the unicast request including the Echo Option, the
   server verifies that the option value equals the stored and

previously sent value; otherwise, the request is silently discarded.
Then, the server verifies that the unicast request has been received
within a pre-configured time interval, as described in
[I-D.ietf-core-echo-request-tag]. In such a case, the request is
further processed and verified; otherwise, it is silently discarded.
Finally, the server updates the Recipient Context associated to that
client, by setting the Replay Window according to the Sequence Number
from the unicast request conveying the Echo Option. The server
either delivers the request to the application if it is an actual
retransmission of the original one, or discards it otherwise.
Mechanisms to signal whether the resent request is a full
retransmission of the original one are out of the scope of this
specification.

In case it does not receive a valid unicast request including the
Echo Option within the configured time interval, the server endpoint
should perform the same challenge-response upon receiving the next
group request from that same client.

A server should not deliver group requests from a given client to the
application until one valid request from that same client has been
verified as fresh, as conveying an echoed Echo Option
[I-D.ietf-core-echo-request-tag]. Also, a server may perform the
challenge-response described above at any time, if synchronization
with sequence numbers of clients is (believed to be) lost, for
instance after a device reboot. It is the role of the application to
define under what circumstances sequence numbers lose
synchronization. This can include a minimum gap between the sequence
number of the latest accepted group request from a client and the
sequence number of a group request just received from the same
client. A client has to be always ready to perform the challenge-
response based on the Echo Option in case a server starts it.

Note that endpoints configured as silent servers are not able to
perform the challenge-response described above, as they do not store
a Sender Context to secure the 4.03 Forbidden response to the client.
Therefore, silent servers should adopt alternative approaches to
achieve and maintain synchronization with sequence numbers of
clients.

This approach provides an assurance of absolute message freshness.
However, it can result in an impact on performance which is
undesirable or unbearable, especially in large groups where many
endpoints at the same time might join as new members or lose
synchronization.

Appendix F. No Verification of Signatures

   There are some application scenarios using group communication that
   have particularly strict requirements. One example of this is the
   requirement of low message latency in non-emergency lighting
   applications [I-D.somaraju-ace-multicast]. For those applications
   which have tight performance constraints and relaxed security
   requirements, it can be inconvenient for some endpoints to verify
   digital signatures in order to assert source authenticity of received
   group messages. In other cases, the signature verification can be
   deferred or only checked for specific actions. For instance, a
   command to turn a bulb on where the bulb is already on does not need
   the signature to be checked. In such situations, the counter
   signature needs to be included anyway as part of the group message,
   so that an endpoint that needs to validate the signature for any
   reason has the ability to do so.

   In this specification, it is NOT RECOMMENDED that endpoints do not
   verify the counter signature of received group messages. However, it
   is recognized that there may be situations where it is not always
   required. The consequence of not doing the signature validation is
   that security in the group is based only on the group-authenticity of
   the shared keying material used for encryption. That is, endpoints
   in the group have evidence that a received message has been
   originated by a group member, although not specifically identifiable
   in a secure way. This can violate a number of security requirements,
   as the compromise of any element in the group means that the attacker
   has the ability to control the entire group. Even worse, the group
   may not be limited in scope, and hence the same keying material might
   be used not only for light bulbs but for locks as well. Therefore,
   extreme care must be taken in situations where the security
   requirements are relaxed, so that deployment of the system will
   always be done safely.

Appendix G. Document Updates

   RFC EDITOR: PLEASE REMOVE THIS SECTION.

G.1. Version -01 to -02

   o Terminology has been made more aligned with RFC7252 and draft-
      ietf-core-object-security: i) "client" and "server" replace the
      old "multicaster" and "listener", respectively; ii) "silent
      server" replaces the old "pure listener".

   o Section 2 has been updated to have the Group Identifier stored in
      the 'ID Context' parameter defined in draft-ietf-core-object-
      security.

o Section 3 has been updated with the new format of the Additional
   Authenticated Data.

o Major rewriting of Section 4 to better highlight the differences
   with the message processing in draft-ietf-core-object-security.

o Added Sections 7.2 and 7.3 discussing security considerations
   about uniqueness of (key, nonce) and collision of group
   identifiers, respectively.

o Minor updates to Appendix A.1 about assumptions on multicast
   communication topology and group size.

o Updated Appendix C on format of group identifiers, with practical
   implications of possible collisions of group identifiers.

o Updated Appendix D.2, adding a pointer to draft-palombini-ace-key-
   groupcomm about retrieval of nodes' public keys through the Group
   Manager.

o Minor updates to Appendix E.3 about Challenge-Response
   synchronization of sequence numbers based on the Echo option from
   draft-ietf-core-echo-request-tag.

G.2. Version -00 to -01

o Section 1.1 has been updated with the definition of group as
   "security group".

o Section 2 has been updated with:

   * Clarifications on etablishment/derivation of security contexts.

   * A table summarizing the the additional context elements
      compared to OSCORE.

o Section 3 has been updated with:

   * Examples of request and response messages.

   * Use of CounterSignature0 rather than CounterSignature.

   * Additional Authenticated Data including also the signature
      algorithm, while not including the Group Identifier any longer.

o Added Section 6, listing the responsibilities of the Group
   Manager.

      o Added Appendix A (former section), including assumptions and
         security objectives.

      o Appendix B has been updated with more details on the use cases.

      o Added Appendix C, providing an example of Group Identifier format.

      o Appendix D has been updated to be aligned with draft-palombini-
         ace-key-groupcomm.

Authors' Addresses

   Marco Tiloca
   RISE SICS
   Isafjordsgatan 22
   Kista   SE-16440 Stockholm
   Sweden

   Email: marco.tiloca@ri.se

   Goeran Selander
   Ericsson AB
   Torshamnsgatan 23
   Kista   SE-16440 Stockholm
   Sweden

   Email: goran.selander@ericsson.com

   Francesca Palombini
   Ericsson AB
   Torshamnsgatan 23
   Kista   SE-16440 Stockholm
   Sweden

   Email: francesca.palombini@ericsson.com

   Jiye Park
   Universitaet Duisburg-Essen
   Schuetzenbahn 70
   Essen   45127
   Germany

   Email: ji-ye.park@uni-due.de