CoRE Working Group M. Tiloca
Internet-Draft RISE AB
Intended status: Standards Track G. Selander
Expires: September 9, 2019 F. Palombini
Ericsson AB
J. Park
Universitaet Duisburg-Essen
March 08, 2019

Group OSCORE - Secure Group Communication for CoAP
draft-ietf-core-oscore-groupcomm-04

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 defines how OSCORE is used in a group communication setting, while fulfilling the same security requirements for group requests and responses. Source authentication of all messages exchanged within the group is provided by means of digital signatures produced by the sender and embedded in the protected CoAP messages.

Status of This Memo

Copyright Notice

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, improve performance and reduce bandwidth utilisation. 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, replay protection and binding of response to request between a sender and a receipient, also in the presence of intermediaries. To this end, a CoAP message is protected by including its payload (if any), certain options, and header fields in a COSE object, which replaces the authenticated and encrypted fields in the protected message.

This document defines Group OSCORE, providing end-to-end security of CoAP messages exchanged between members of a group, and preserving independence of transport layer. In particular, the described approach defines how OSCORE should be used in a group communication

setting, so that end-to-end security is assured in the same way as OSCORE for unicast communication. That is, end-to-end security is provided for CoAP multicast requests sent by a client to the group, and for related CoAP responses sent by multiple servers. 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 [RFC6347] to protect hop-by-hop communication between a sender and a proxy (and vice versa), and between a proxy and a recipient (and vice versa). Note that DTLS cannot be used to secure messages sent over multicast.

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 security parameters (Common Context, see Section 2). The term group used in this specification refers thus to a "security group", not to be confused with network/multicast group or application group.

- Group Manager: entity responsible for a group. Each endpoint in a group communicates securely with the respective Group Manager, which is neither required to be an actual group member nor to take part in the group communication. The full list of responsibilities of the Group Manager is provided in Section 7.

- Silent server: member of a group that never responds to requests. Note that a silent server can act as a client, the two roles are independent.

- Group Identifier (Gid): identifier assigned to the group. Group Identifiers should be unique within the set of groups of a given Group Manager, in order to avoid collisions. In case they are not, the considerations in Section 8.5 apply.

- Group request: CoAP request message sent by a client in the group to all servers in that group.

- Source authentication: evidence that a received message in the group originated from a specific identified group member. This also provides assurance that the message was not tampered with by anyone, be it a different legitimate group member or an endpoint which is not a 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], extended as defined below. Each endpoint in a group makes use of:

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

   * The ID Context parameter contains the Gid of the group, which is used to retrieve the Security Context for processing messages intended to the endpoints of the group (see Section 6). The choice of the Gid is application specific. An example of specific formatting of the Gid is given in Appendix C. The application needs to specify how to handle possible collisions between Gids, see Section 8.5.

   * A new parameter Counter Signature Algorithm is included. Its value identifies the digital signature algorithm used to compute a counter signature on the COSE object (see Section 4.5 of [RFC8152]) which provides source authentication within the group. Its value is immutable once the Common Context is established. The used Counter Signature Algorithm

MUST be selected among the signing ones defined in the COSE Algorithms Registry (see section 16.4 of [RFC8152]). The EdDSA signature algorithm ed25519 [RFC8032] is mandatory to implement. If Elliptic Curve Digital Signature Algorithm (ECDSA) is used, it is RECOMMENDED that implementations implement "deterministic ECDSA" as specified in [RFC6979].

* A new parameter Counter Signature Parameters is included. This parameter identifies the parameters associated to the digital signature algorithm specified in the Counter Signature Algorithm. This parameter MAY be empty and is immutable once the Common Context is established. The exact structure of this parameter depends on the value of Counter Signature Algorithm, and is defined in the Counter Signature Parameters Registry (see Section 9.1), where each entry indicates a specified structure of the Counter Signature Parameters.

2. one Sender Context, unless the endpoint is configured exclusively as silent server. The Sender Context is used to secure outgoing messages and is initialized according to Section 3 of [I-D.ietf-core-object-security], once the endpoint has joined the group. The Sender Context of a given endpoint matches the corresponding Recipient Context in all the endpoints receiving a protected message from that endpoint. Besides, in addition to what is defined in [I-D.ietf-core-object-security], the Sender Context stores also the endpoint's private key.

3. one Recipient Context for each distinct endpoint from which messages are received, used to process incoming messages. The recipient may generate the Recipient Context upon receiving an incoming message from another endpoint in the group for the first time (see Section 6.2 and Section 6.4). Each Recipient Context matches the Sender Context of the endpoint from which protected 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 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
Common Context	Counter signature parameters
Sender Context	Endpoint's own private 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 uses the sender's public key, in order to verify the counter signature of the COSE Object (see Section 3).

If not already stored in the Recipient Context associated to the sender, the recipient retrieves the sender's public key from the Group Manager, which collects public keys upon endpoints' joining the group, acts as trusted key repository and ensures the correct association between the public key and the identifier of the sender, for instance by means of public key certificates.

Note that a group member can retrieve public keys from the Group Manager and generate the Recipient Context associated to another group member at any point in time, as long as this is done before verifying a received secure CoAP message. The exact configuration is application dependent. For example, an application can configure a group member to retrieve all the required information and to create the Recipient Context exactly upon receiving a message from another group member for the first time. As an alternative, the application can configure a group member to asynchronously retrieve the required information and update its list of Recipient Contexts well before receiving any message, e.g. by Observing [RFC7641] the Group Manager to get updates on the group membership.

It is RECOMMENDED that the Group Manager collects public keys and provides them to group members upon request as described in [I-D.ietf-ace-key-groupcomm-oscore], where the join process is based on the ACE framework for Authentication and Authorization in constrained environments [I-D.ietf-ace-oauth-authz]. Further details about how public keys can be handled and retrieved in the group is out of the scope of this document.

An endpoint receives its own Sender ID from the Group Manager upon joining the group. That Sender ID is valid only within that group, and is unique within the group. An endpoint uses its own Sender ID (together with other data) to generate unique AEAD nonces for outgoing messages, as in [I-D.ietf-core-object-security]. Endpoints which are configured only as silent servers do not have a Sender ID.

The Sender/Recipient Keys and the Common IV are derived according to the same scheme defined in Section 3.2 of [I-D.ietf-core-object-security]. The mandatory-to-implement HKDF and AEAD algorithms for Group OSCORE are the same as in [I-D.ietf-core-object-security].

2.1. Management of Group Keying Material

In order to establish a new Security Context in a group, a new Group Identifier (Gid) for that group and a new value for the Master Secret parameter MUST be distributed. An example of Gid 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 Gid.

After a new Gid has been distributed, a same Recipient ID ('kid') should not be considered as a persistent and reliable indicator of the same group member. Such an indication can be actually achieved only by verifying countersignatures of received messages.

As a consequence, group members may end up retaining stale Recipient Contexts, that are no longer useful to verify incoming secure messages. Applications may define policies to delete (long-)unused Recipient Contexts and reduce the impact on storage space.

If the application requires so (see Appendix A.1), it is RECOMMENDED to adopt a group key management scheme, and securely distribute a new value for the Gid and 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 to preserve backward security and forward security in the group, if the application requires it.

The specific approach used to distribute the new Gid and Master Secret parameter to the group is out of the scope of this document. However, it is RECOMMENDED that the Group Manager supports the distribution of the new Gid and Master Secret parameter to the group according to the Group Rekeying Process described in [I-D.ietf-ace-key-groupcomm-oscore].

2.2. Wrap-Around of Partial IVs

A client can eventually experience a wrap-around of its own Sender Sequence Number, which is used as Partial IV in outgoing requests and incremented after each request.

When this happens, the endpoint MUST NOT transmit further group requests until it has derived a new Sender Context, in order to avoid reusing nonces with the same keys.

Furthermore, the endpoint SHOULD inform the Group Manager, that can take one of the following actions:

o The Group Manager renews the OSCORE Security Context in the group (see Section 2.1).

o The Group Manager provides a new Sender ID value to the endpoint that has experienced the wrap-around. Then, the endpoint derives a new Sender Context using the new Sender ID, as described in Section 3.2 of [I-D.ietf-core-object-security].

Either case, same considerations from Section 2.1 hold about possible retaining of stale Recipient Contexts.

3. The COSE Object

Building on Section 5 of [I-D.ietf-core-object-security], this section defines how to use COSE [RFC8152] to wrap and protect data in the original message. OSCORE uses the untagged COSE_Encrypt0 structure with an Authenticated Encryption with Additional Data (AEAD) algorithm. For Group OSCORE, the following modifications apply.

3.1. Updated external_aad

The external_aad in the Additional Authenticated Data (AAD) is extended with the counter signature algorithm and related parameters used to sign messages. In particular, compared with Section 5.4 of [I-D.ietf-core-object-security], the 'algorithms' array in the aad_array MUST also include:

o 'alg_countersign', which contains the Counter Signature Algorithm from the Common Context (see Section 2). This parameter has the value specified in the "Value" field of the Counter Signature Parameters Registry (see Section 9.1) for this counter signature algorithm.

The 'algorithms' array in the aad_array MAY also include:

- 'par_countersign', which contains the Counter Signature Parameters from the Common Context (see Section 2). This parameter contains the counter signature parameters encoded as specified in the "Parameters" field of the Counter Signature Parameters Registry (see Section 9.1), for the used counter signature algorithm. Note that if the Counter Signature Parameters in the Common Context is empty, 'par_countersign' is not present.

This external_aad structure is used both for the encryption process producing the ciphertext (see Section 5.3 of [RFC8152]) and for the signing process producing the counter signature, as defined below.

```
external_aad = bstr .cbor aad_array

aad_array = [
   oscore_version : uint,
   algorithms : [alg_aead : int / tstr ,
                 alg_countersign : int / tstr ,
                 ? par_countersign : any],
   request_kid : bstr,
   request_piv : bstr,
   options : bstr
]
```

3.2. Use of the 'kid' Parameter

The value of the 'kid' parameter in the 'unprotected' field of response messages MUST be set to the Sender ID of the endpoint transmitting the message. That is, unlike in [I-D.ietf-core-object-security], the 'kid' parameter is always present in all messages, i.e. both requests and responses.

3.3. Updated 'unprotected' Field

The 'unprotected' field MUST additionally include the following parameter:

- CounterSignature0 : its value is set to the counter signature of the COSE object, computed by the sender using its own private key as described in Appendix A.2 of [RFC8152]. In particular, the Sig_structure contains the external_aad as defined above and the ciphertext of the COSE_Encrypt0 object as payload.

4. OSCORE Header Compression

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 and the OSCORE Payload.

4.1. Encoding of the OSCORE Option Value

Analogously to [I-D.ietf-core-object-security], the value of the OSCORE option SHALL contain the OSCORE flag bits, the Partial IV parameter, the kid context parameter (length and value), and the kid parameter, with the following modifications:

- o The first byte, containing the OSCORE flag bits, has the following encoding modifications:

  * The fourth least significant bit MUST be set to 1 in every message, to indicate the presence of the 'kid' parameter for all group requests and responses. That is, unlike in [I-D.ietf-core-object-security], the 'kid' parameter is always present in all messages.

  * The fifth least significant bit MUST be set to 1 for group requests, to indicate the presence of the 'kid context' parameter in the compressed COSE object. The 'kid context' MAY be present in responses if the application requires it. In such a case, the kid context flag MUST be set to 1.

The flag bits are registered in the OSCORE Flag Bits registry specified in Section 13.7 of [I-D.ietf-core-object-security].

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

- o 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 ------------>
+-+-+-+-+-+-+-+-+-----------------------------------+
|0 0|0|h|1|  n  |       Partial IV (if any)         |
+-+-+-+-+-+-+-+-+-----------------------------------+

 <-- 1 byte ---> <------- s bytes ------>
+---------------+-----------------------+-----------+
|  s (if any)   |   kid context = Gid   |    kid    |
+---------------+-----------------------+-----------+
```

Figure 2: OSCORE Option Value

4.2. Encoding of the OSCORE Payload

The payload of the OSCORE message SHALL encode the ciphertext of the COSE object concatenated with the value of the CounterSignature0 of the COSE object, computed as in Appendix A.2 of [RFC8152] according to the Counter Signature Algorithm and Counter Signature Parameters in the Security Context.

4.3. Examples of Compressed COSE Objects

This section covers a list of OSCORE Header Compression examples for group requests and responses. The examples assume that the COSE_Encrypt0 object is set (which means the CoAP message and cryptographic material is known). Note that the examples do not include the full CoAP unprotected message or the full security context, but only the input necessary to the compression mechanism, i.e. the COSE_Encrypt0 object. The output is the compressed COSE object as defined in Section 4 and divided into two parts, since the object is transported in two CoAP fields: OSCORE option and payload.

The examples assume that 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.

1. Request with ciphertext = 0xaea0155667924dff8a24e4cb35b9, kid = 0x25, Partial IV = 5 and kid context = 0x44616c

Before compression (96 bytes):

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

After compression (85 bytes):

```
Flag byte: 0b00011001 = 0x19

Option Value: 19 05 03 44 61 6c 25 (7 bytes)

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

1. Response with ciphertext = 60b035059d9ef5667c5a0710823b, kid = 0x52 and no Partial IV.

```
Before compression (88 bytes):

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

After compression (80 bytes):

Flag byte: 0b00001000 = 0x08

Option Value: 08 52 (2 bytes)

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

5. Message Binding, Sequence Numbers, Freshness and Replay Protection

The requirements and properties described in Section 7 of [I-D.ietf-core-object-security] also apply to OSCORE used in group communication. In particular, group OSCORE provides message binding of responses to requests, which provides relative freshness of responses, and replay protection of requests.

Besides, group OSCORE provides additional assurances on the client side, upon receiving responses bound to a same request. That is, as long as the client retains the CoAP Token used in a request (see Section 2.5 of [RFC7390]), group OSCORE ensures that: any possible response sent to that request is not a replay; and at most one response to that request from a given server is accepted, if required by the application.

More details about error processing for replay detection in group OSCORE are specified in Section 6 of this specification. The mechanisms describing replay protection and freshness of Observe notifications do not apply to group OSCORE, as Observe is not defined for group settings.

5.1. Synchronization of Sender Sequence Numbers

Upon joining the group, new servers are not aware of the Sender 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 or (purposely) delayed. The same holds when a server loses

synchronization with Sender Sequence Numbers of clients, for instance after a device reboot.

The exact way to address this issue is application specific, and depends on the particular use case and its synchronization requirements. The list of methods to handle synchronization of Sender 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 for synchronization of sequence numbers.

6. 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.

As per [RFC7252][RFC7390], group requests sent over multicast MUST be Non-Confirmable. Thus, senders should store their outgoing messages for an amount of time defined by the application and sufficient to correctly handle possible retransmissions. However, this does not prevent the acknowledgment of Confirmable group requests in non-multicast environments. Besides, according to Section 5.2.3 of [RFC7252], responses to Non-Confirmable group requests SHOULD be also Non-Confirmable. However, endpoints MUST be prepared to receive Confirmable responses in reply to a Non-Confirmable group request.

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 recipient MUST stop processing and silently reject any message which is malformed and does not follow the format specified in Section 3, or which is not cryptographically validated in a successful way. Either case, it is RECOMMENDED that the recipient does not send 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.

6.1. 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 encryption of the COSE object is modified as described in Section 3. The encoding of the compressed COSE object is modified as described in Section 4.

- o In step 5, the counter signature is computed and the format of the OSCORE mesage is modified as described in Section 4.2. In particular, the payload of the OSCORE message includes also the counter signature.

6.2. 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 follows Section 4. If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the server creates a new Recipient Context, initializes it according to Section 3 of [I-D.ietf-core-object-security], also retrieving 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.

- o Additionally, if the used Recipient Context was created upon receiving this group request and the message is not verified successfully, the server MAY delete that Recipient Context. Such a configuration, which is specified by the application, would prevent attackers from overloading the server's storage and creating processing overhead on the server.

6.3. 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.

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

- In step 5, the counter signature is computed and the format of the OSCORE mesage is modified as described in Section 4.2. In particular, the payload of the OSCORE message includes also the counter signature.

6.4. 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.

- In step 2, the decoding of the compressed COSE object is modified as described in Section 3. The client also checks whether it previously received a secure response to this request, such that it was successfully verified and included the same Recipient ID ('kid') of the just received response. In case of positive match the client SHALL stop processing the response. If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the client creates a new Recipient Context, initializes it according to Section 3 of [I-D.ietf-core-object-security], also retrieving the server's public key.

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

- In step 5, the client also verifies the counter signature using the public key of the server from the associated Recipient Context. In case of success, the client also records the received Recipient ID ('kid') as included in a successfully verified response to the request.

- Additionally, if the used Recipient Context was created upon receiving this response and the message is not verified successfully, the client MAY delete that Recipient Context. Such a configuration, which is specified by the application, would prevent attackers from overloading the client's storage and creating processing overhead on the client.

Upon freeing up the Token value of a secure group request for possible reuse [RFC7390], the client MUST delete the list of recorded Recipient IDs associated to that request (see step 5 above).

7. Responsibilities of the Group Manager

The Group Manager is responsible for performing the following tasks:

1. Creating and managing OSCORE groups. This includes the assignment of a Gid to every newly created group, as well as ensuring uniqueness of Gids within the set of its OSCORE groups.

2. Defining policies for authorizing the joining of its OSCORE groups. Such policies can be enforced locally by the Group Manager, or by a third party in a trust relation with the Group Manager and entrusted to enforce join policies on behalf of the Group Manager.

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

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

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

6. Defining a communication policy for each of its OSCORE groups, and signalling it to new endpoints during the join process.

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

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

9. Updating the Gid of its OSCORE groups, upon renewing the respective Security Context.

10. Acting as key repository, in order to handle the public keys of the members of its OSCORE groups, and providing such public keys to other members of the same group upon request. The actual storage of public keys may be entrusted to a separate secure storage device.

8. 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.

8.1. Group-level Security

The approach described in this document relies on commonly shared group keying material to protect communication within a group. This has the following implications.

- o 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.

- o The AEAD algorithm provides only group authentication, i.e. it ensures that a message sent to a group has been sent by a member of that group, but not by the alleged sender. This is why source authentication of messages sent to a group is ensured through a counter signature, which is computed by the sender using its own private key and then appended to the message payload.

Note that, even if an endpoint is authorized to be a group member and to take part in group communications, there is a risk that it behaves inappropriately. For instance, it can forward the content of messages in the group 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), which results in a practically limited risk and enables a prompt detection/reaction in case of misbehaving.

8.2. 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 The case A in Appendix D.3 of [I-D.ietf-core-object-security] for messages including a Partial IV concerns only group requests, and same considerations from [I-D.ietf-core-object-security] apply here as well.

- o The case B in Appendix D.3 of [I-D.ietf-core-object-security] for messages not including a Partial IV concerns all group responses, and same considerations from [I-D.ietf-core-object-security] apply here as well.

As a consequence, 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.

8.3. Management of Group Keying Material

The approach described in this specification should take into account the risk of compromise of group members. In particular, this document specifies that a key management scheme for secure revocation and renewal of Security Contexts and group keying material should be adopted.

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.

8.4. Update of Security Context and Key Rotation

A group member can receive a message shortly after the group has been rekeyed, and new security parameters and keying material have been distributed by the Group Manager. In the following two cases, this may result in misaligned Security Contexts between the sender and the recipient.

In the first case, the sender protects a message using the old Security Context, i.e. before having installed the new Security Context. However, the recipient receives the message after having installed the new Security Context, hence not being able to correctly process it. A possible way to ameliorate this issue is to preserve the old, recent, Security Context for a maximum amount of time defined by the application. By doing so, the recipient can still try to process the received message using the old retained Security Context as second attempt. Note that a former (compromised) group member can take advantage of this by sending messages protected with the old retained Security Context. Therefore, a conservative application policy should not admit the storage of old Security Contexts.

In the second case, the sender protects a message using the new Security Context, but the recipient receives that request before having installed the new Security Context. Therefore, the recipient would not be able to correctly process the request and hence discards it. If the recipient receives the new Security Context shortly after that and the sender endpoint uses CoAP retransmissions, the former will still be able to receive and correctly process the message. In any case, the recipient should actively ask the Group Manager for an updated Security Context according to an application-defined policy, for instance after a given number of unsuccessfully decrypted incoming messages.

8.5. 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. That can also happen if the application can not guarantee unique Group Identifiers within a given Group Manager. 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, AEAD keys are different among different groups.

9. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[This Document]" with the RFC number of this specification and delete this paragraph.

This document has the following actions for IANA.

9.1. Counter Signature Parameters Registry

This specification establishes the IANA "Counter Signature Parameters" Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 9.2.

The columns of this table are:

- Name: A value that can be used to identify an algorithm in documents for easier comprehension. Its value is taken from the 'Name' column of the "COSE Algorithms" Registry.

- o Value: The value to be used to identify this algorithm. Its content is taken from the 'Value' column of the "COSE Algorithms" Registry. The value MUST be the same one used in the "COSE Algorithms" Registry for the entry with the same 'Name' field.

- o Parameters: This indicates the CBOR encoding of the parameters (if any) for the counter signature algorithm indicated by the 'Value' field.

- o Description: A short description of the parameters encoded in the 'Parameters' field (if any).

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

Initial entries in the registry are as follows.

Name	Value	Parameters	Description	Reference
EdDSA	-8	crv : int	crv value taken from the COSE Elliptic Curve Registry	[This Document]
ES256	-7	crv : int	crv value taken from the COSE Elliptic Curve Registry	[This Document]
ES384	-35	crv : int	crv value taken from the COSE Elliptic Curve Registry	[This Document]
ES512	-36	crv : int	crv value taken from the COSE Elliptic Curve Registry	[This Document]

PS256	-37		Parameters not present	[This Document]
PS384	-38		Parameters not present	[This Document]
PS512	-39		Parameters not present	[This Document]
RSAES-OAEP w/ RFC 8017 default parameters	-40		Parameters not present	[This Document]
RSAES-OAEP w/ SHA-256	-41		Parameters not present	[This Document]
RSAES-OAEP w/ SHA-512	-42		Parameters not present	[This Document]

9.2. Expert Review Instructions

The IANA Registry established in this document is defined as expert review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

TBD

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-16 (work in progress), March 2019.

[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>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

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

10.2. Informative References

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-00 (work in progress), December 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-22 (work in progress), March 2019.

[I-D.ietf-core-echo-request-tag]
Amsuess, C., Mattsson, J., and G. Selander, "Echo and Request-Tag", draft-ietf-core-echo-request-tag-03 (work in progress), October 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.

[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>.

[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>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

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.

- 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.

- 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.

- o Communication with the Group Manager: an endpoint must use a secure dedicated channel when communicating with the Group Manager, also when not registered as group member.

- o Provisioning and management of Security Contexts: an OSCORE Security Context must be established among the group members. 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 provisioning and management of the Security Context is out of the scope of this document.

- 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.

- 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.

- 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.

- 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.

- Message ordering: it must be possible to determine the ordering of messages coming from a single sender. 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 senders.

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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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'.

Using an immutable Group Prefix for a group assumes that enough time elapses between two consecutive usages of the same Group Epoch value in that group. This ensures that the Gid value is temporally unique during the lifetime of a given message. Thus, the expected highest rate for addition/removal of group members and consequent group rekeying should be taken into account for a proper dimensioning of the Group Epoch size.

As discussed in Section 8.5, 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 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 Identifiers 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. When becoming members of a group, endpoints are not required to know how many and what endpoints are in the same group.

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 the scope of this document.

The Group Manager must verify that the joining endpoint is authorized to join 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. Further details about the authorization of joining endpoints are out of scope.

In case of successful authorization check, the Group Manager generates a Sender ID assigned to the joining endpoint, before proceeding with the rest of the join process. That is, the Group Manager provides the joining endpoint with the keying material and parameters to initialize the OSCORE Security Context (see Section 2). The actual provisioning of keying material and parameters to the joining endpoint is out of the scope of this document.

It is RECOMMENDED that the join process adopts the approach described in [I-D.ietf-ace-key-groupcomm-oscore] and based on the ACE framework for Authentication and Authorization in constrained environments [I-D.ietf-ace-oauth-authz].

Appendix E. Examples of Synchronization Approaches

This section describes three possible approaches that can be considered by server endpoints to synchronize with sender 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 sender 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 sender 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 or delayed 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 according to 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 6.2 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 sender 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 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 sender 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 sender sequence numbers lose synchronization. This can include a minimum gap between the sender sequence number of the latest accepted group request from a client and the sender 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 sender 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 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 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 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 -03 to -04

- Added the new "Counter Signature Parameters" in the Security Common Context (see Section 2).

- Added recommendation on using "deterministic ECDSA" if ECDSA is used as counter signature algorithm (see Section 2).

- Clarified possible asynchronous retrieval of key material from the Group Manager, in order to process incoming messages (see Section 2).

- Structured Section 3 into subsections.

- Added the new 'par_countersign' to the aad_array of the external_aad (see Section 3.1).

- Clarified non reliability of 'kid' as identity indicator for a group member (see Section 2.1).

- Described possible provisioning of new Sender ID in case of Partial IV wrap-around (see Section 2.2).

- The former signature bit in the Flag Byte of the OSCORE option value is reverted to reserved (see Section 4.1).

- Updated examples of compressed COSE object, now with the sixth less significant bit in the Flag Byte of the OSCORE option value set to 0 (see Section 4.3).

- Relaxed statements on sending error messages (see Section 6).

- Added explicit step on computing the counter signature for outgoing messages (see Setions 6.1 and 6.3).

- Handling of just created Recipient Contexts in case of unsuccessful message verification (see Sections 6.2 and 6.4).

- Handling of replied/repeated responses on the client (see Section 6.4).

- New IANA Registry "Counter Signature Parameters" (see Section 9.1).

G.2. Version -02 to -03

- Revised structure and phrasing for improved readability and better alignment with draft-ietf-core-object-security.

- Added discussion on wrap-Around of Partial IVs (see Section 2.2).

- Separate sections for the COSE Object (Section 3) and the OSCORE Header Compression (Section 4).

- The countersignature is now appended to the encrypted payload of the OSCORE message, rather than included in the OSCORE Option (see Section 4).

- Extended scope of Section 5, now titled " Message Binding, Sequence Numbers, Freshness and Replay Protection".

- Clarifications about Non-Confirmable messages in Section 5.1 "Synchronization of Sender Sequence Numbers".

- Clarifications about error handling in Section 6 "Message Processing".

- Compacted list of responsibilities of the Group Manager in Section 7.

- Revised and extended security considerations in Section 8.

- Added IANA considerations for the OSCORE Flag Bits Registry in Section 9.

- Revised Appendix D, now giving a short high-level description of a new endpoint set-up.

G.3. Version -01 to -02

- 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".

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

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

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

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

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

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

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

- 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.4. Version -00 to -01

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

- Section 2 has been updated with:

  * Clarifications on etablishment/derivation of security contexts.

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

- 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.

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

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

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

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

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

Acknowledgments

The authors sincerely thank Stefan Beck, Rolf Blom, Carsten Bormann, Esko Dijk, Klaus Hartke, Rikard Hoeglund, 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 VINNOVA and the Celtic-Next project CRITISEC; and by the EIT-Digital High Impact Initiative ACTIVE.

Authors' Addresses

Marco Tiloca
RISE AB
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