CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 8, 2020

M. Tiloca
RISE AB
G. Selander
F. Palombini
Ericsson AB
J. Park
Universitaet Duisburg-Essen

April 06, 2020

Group OSCORE - Secure Group Communication for CoAP
draft-ietf-core-oscore-groupcomm-08

Abstract

This document defines Group Object Security for Constrained RESTful Environments (Group OSCORE), providing end-to-end security of CoAP messages exchanged between members of a group, e.g. using IP multicast. In particular, the described approach defines how OSCORE should be used in a group communication setting to provide source authentication for CoAP group requests, sent by a client to multiple servers, and the corresponding CoAP responses.

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 October 8, 2020.

Table of Contents

1. Introduction 4
1.1. Terminology 5
2. Security Context 7
2.1. Common Context 7
2.2. Sender Context and Recipient Context 8
2.3. The Group Manager 9
2.4. Management of Group Keying Material 10
2.5. Exhaustion of Partial IV Values 11
3. Pairwise Keys 12
3.1. Key Derivation 12
3.2. Usage of Sequence Numbers 13
3.3. Note on Implementation 13
4. The COSE Object 14
4.1. Counter Signature 14
4.2. The 'kid' and 'kid context' parameters 14
4.3. external_aad 15
4.3.1. external_aad for Encryption 15
4.3.2. external_aad for Signing 16
5. OSCORE Header Compression 17
5.1. Examples of Compressed COSE Objects 17
6. Message Binding, Sequence Numbers, Freshness and Replay Protection 19
6.1. Synchronization of Sender Sequence Numbers 19
7. Message Processing 19
7.1. Protecting the Request 20
7.1.1. Supporting Observe 21
7.2. Verifying the Request 21
7.2.1. Supporting Observe 22
7.3. Protecting the Response 22
7.3.1. Supporting Observe 23
7.4. Verifying the Response 23
7.4.1. Supporting Observe 24
8. Responsibilities of the Group Manager 24
9. Optimized Mode 25
9.1. Optimized Request 25
9.1.1. Optimized Compressed Request 25
9.2. Optimized Response 26
9.2.1. Optimized Compressed Response 26
10. Security Considerations 26

10.1. Group-level Security 27
10.2. Uniqueness of (key, nonce) 28
10.3. Management of Group Keying Material 28
10.4. Update of Security Context and Key Rotation 29
10.4.1. Late Update on the Sender 29
10.4.2. Late Update on the Recipient 30
10.5. Collision of Group Identifiers 30
10.6. Cross-group Message Injection 30
10.7. Group OSCORE for Unicast Requests 32
10.8. End-to-end Protection 33
10.9. Security Context Establishment 33
10.10. Master Secret 34
10.11. Replay Protection 34
10.12. Client Aliveness 35
10.13. Cryptographic Considerations 35
10.14. Message Segmentation 36
10.15. Privacy Considerations 36
11. IANA Considerations 37
11.1. Counter Signature Parameters Registry 37
11.2. Counter Signature Key Parameters Registry 40
11.3. OSCORE Flag Bits Registry 42
11.4. Expert Review Instructions 42
12. References 43
12.1. Normative References 43
12.2. Informative References 44
Appendix A. Assumptions and Security Objectives 46
A.1. Assumptions 47
A.2. Security Objectives 48
Appendix B. List of Use Cases 49
Appendix C. Example of Group Identifier Format 51
Appendix D. Set-up of New Endpoints 52
Appendix E. Examples of Synchronization Approaches 53
E.1. Best-Effort Synchronization 53
E.2. Baseline Synchronization 53
E.3. Challenge-Response Synchronization 54
Appendix F. No Verification of Signatures 56
Appendix G. Pairwise Mode 57
G.1. Pre-Requirements 57
G.2. Pairwise Protected Request 57
G.3. Pairwise Protected Response 58
Appendix H. Document Updates 58
H.1. Version -07 to -08 58
H.2. Version -06 to -07 60
H.3. Version -05 to -06 60
H.4. Version -04 to -05 61
H.5. Version -03 to -04 61
H.6. Version -02 to -03 62
H.7. Version -01 to -02 63

H.8. Version -00 to -01 64
Acknowledgments 64
Authors' Addresses 65

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 [I-D.ietf-core-groupcomm-bis] addresses use cases where deployed devices benefit from a group communication model, for example to reduce latencies, improve performance and reduce bandwidth utilization. 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). This specification defines the security protocol for Group communication for CoAP [I-D.ietf-core-groupcomm-bis].

Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] 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 recipient, independent of transport 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 the same end-to-end security properties as OSCORE in the case where CoAP requests have multiple recipients. In particular, the described approach defines how OSCORE should be used in a group communication setting to provide source authentication for CoAP group requests, sent by a client to multiple servers, and the corresponding CoAP responses.

Group OSCORE provides source authentication of CoAP messages, by means of two possible methods. The first method relies on a digital signature produced with the private key of the sender and embedded in the protected CoAP message. The second method relies on a symmetric key, which is derived from a pairwise shared secred computed from the asymmetric keys of the message sender and recipient.

The second method is intended for one-to-one messages sent in the group. These include all responses, as individually sent by servers, as well as requests addressed to an individual server. For instance, such requests are sent as part of an exchange using the CoAP Echo

option [I-D.ietf-core-echo-request-tag], or as part of a block-wise transfer [RFC7959] in the group, after the first block-wise request possibly targeting all servers in the group and including the CoAP Block2 option (see Section 2.3.6 of [I-D.ietf-core-groupcomm-bis]).

Just like OSCORE, Group OSCORE is independent of transport layer and works wherever CoAP does. Group communication for CoAP [I-D.ietf-core-groupcomm-bis] uses UDP/IP multicast as the underlying data transport.

As with OSCORE, it is possible to combine Group OSCORE with communication security on other layers. One example is the use of transport layer security, such as DTLS [RFC6347], between one client and one proxy (and vice versa), or between one proxy and one server (and vice versa), in order to protect the routing information of packets from observers. Note that DTLS cannot be used to secure messages sent over IP multicast.

Group OSCORE defines different modes of operation:

- In the signature mode, Group OSCORE requests and responses are digitally signed. The signature mode supports all COSE algorithms as well as signature verification by intermediaries.

- The pairwise mode allows two group members to exchange unicast OSCORE requests and responses protected with symmetric keys. These symmetric keys are derived from Diffie-Hellman shared secrets, calculated with the asymmetric keys of the two group members. This allows for shorter integrity tags and therefore lower message overhead.

- In the (hybrid) optimized mode, the CoAP requests are digitally signed as in the signature mode, and the CoAP responses are integrity protected with the symmetric key of the pairwise mode.

The signature and optimized modes are detailed in the body of this document. The pairwise mode is detailed in Appendix G. Unless otherwise stated, this specification refers to the signature mode.

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 [I-D.ietf-core-groupcomm-bis]; 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 [RFC8613].

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

This document refers also to the following terminology.

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

- Group: a set of endpoints that share group keying material and security parameters (Common Context, see Section 2). Unless specified otherwise, the term group used in this specification refers thus to a "security group", not to be confused with CoAP/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 8.

- Silent server: member of a group that never responds to requests. Given that, for CoAP group communications, messages are normally sent without requesting a confirmation, the idea of a server silently acting on a message is not unreasonable. Note that an endpoint can implement both a silent server and a client, the two roles are independent. An endpoint implementing only a silent server processes only incoming requests, and, in case it supports only the signature mode, it maintains less keying material and especially does not have a Sender Context for the group.

- Group Identifier (Gid): identifier assigned to the group. Group Identifiers must be unique within the set of groups of a given Group Manager.

- 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. Security Context

Each endpoint registered as member of a group maintains a Security Context as defined in Section 3 of [RFC8613], extended as follows (see Figure 1):

- One Common Context, shared by all the endpoints in the group. Three new parameters are included in the Common Context: Counter Signature Algorithm, Counter Signature Parameters and Counter Signature Key Parameters.

- One Sender Context, extended with the endpoint's private key. The Sender Context is omitted if the endpoint is configured exclusively as silent server.

- One Recipient Context for each endpoint from which messages are received. No Recipient Contexts are maintained as associated to endpoints from which messages are not (expected to be) received. The Recipient Context is extended with the public key of the associated endpoint.

Context Component	New Information Elements
Common Context	Counter Signature Algorithm Counter Signature Parameters Counter Signature Key Parameters
Sender Context	Endpoint's own private key
Each Recipient Context	Public key of the other endpoint

Figure 1: Additions to the OSCORE Security Context

2.1. Common Context

The ID Context parameter (see Sections 3.3 and 5.1 of [RFC8613]) in the Common Context SHALL contain the Group Identifier (Gid) of the group. 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 10.5.

The Counter Signature Algorithm identifies the digital signature algorithm used to compute a counter signature on the COSE object (see Section 4.5 of [RFC8152]). 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].

The Counter Signature Parameters 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 11.1), where each entry indicates a specified structure of the Counter Signature Parameters.

The Counter Signature Key Parameters identifies the parameters associated to the keys used with 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 Key Parameters Registry (see Section 11.2), where each entry indicates a specified structure of the Counter Signature Key Parameters.

2.2. Sender Context and Recipient Context

OSCORE specifies the derivation of Sender Context and Recipient Context, specifically Sender/Recipient Keys and Common IV, from a set of input parameters (see Section 3.2 of [RFC8613]). This derivation applies also to Group OSCORE, and the mandatory-to-implement HKDF and AEAD algorithms are the same as in [RFC8613]. However, for Group OSCORE the Sender Context and Recipient Context additionally contain asymmetric keys.

The Sender Context needs to be configured with the private key of the endpoint. The private key is used to generate a signature (see Section 4) included in the sent OSCORE message. How the private key is established is out of scope for this specification.

Each Recipient Context needs to be configured with the public key of the associated endpoint. The public key is used to verify a signature (see Section 4) included in the received OSCORE message.

The input parameters for deriving the Recipient Context parameters and the public key of the associated endpoint may be provided to the recipient endpoint upon joining the group. These parameters may alternatively be acquired at a later time, for example the first time a message is received from this particular endpoint in the group (see Section 7.2 and Section 7.4). The received message together with the Common Context contains the necessary information to derive a security context for verifying a message, except for the public key of the associated endpoint.

For severely constrained devices, it may be not feasible to simultaneously handle the ongoing processing of a recently received message in parallel with the retrieval of the associated endpoint's public key. Such devices can be configured to drop a received message for which there is no Recipient Context, and retrieve the public key in order to have it available to verify subsequent messages from that endpoint.

2.3. The Group Manager

Endpoints communicating with Group OSCORE need, in addition to the OSCORE input parameters, also to be provisioned with information about the group(s) and other endpoints in the group(s).

The Group Manager is an entity responsible for the group, including the Group Identifier (Gid) used as ID Context, as well as the Sender ID and Recipient ID of the group members (see Section 8).

The Group Manager is exclusively in control of the Gid values uniquely assigned to the different groups under its control, as well as of the Sender ID and Recipient ID values uniquely assigned to the members of each of those groups. According to a hierarchical approach, the Gid value assigned to a group is associated to a dedicated space for the values of Sender ID and Recipient ID of the members of that group. In addition, the Group Manager records the public keys of endpoints joining a group, and provides information about the group and its members to other members.

An endpoint receives the Group Identifier and OSCORE input parameters, including 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. Endpoints which are configured only as silent servers do not have a Sender ID.

A group member can retrieve from the Group Manager the public key and other information associated to another group member, with which it can generate the corresponding Recipient Context. An application can configure a group member to asynchronously retrieve information about Recipient Contexts, e.g. by Observing [RFC7641] the Group Manager to get updates on the group membership.

According to this specification, it is RECOMMENDED to use a Group Manager 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.

The Group Manager may serve additional entities acting as signature checkers, e.g. intermediary gateways. These entities do not join a group as members, but can retrieve public keys of group members from the Group Manager, in order to verify counter signatures of group messages. A signature checker is required to be authorized for retrieving public keys of members in a specific group from the Group Manager. To this end, the same method mentioned above and based on the ACE framework can be used.

2.4. 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. When distributing the new Gid and Master Secret, the Group Manager MAY distribute also a new value for the Master Salt parameter, and SHOULD preserve the current value of the Sender ID of each group member.

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 and Master Secret parameter. The Master Salt used for the re-derivations is the updated Master Salt parameter if provided by the Group Manager, or the empty byte string otherwise. From then on, each group member MUST use its latest installed Sender Context to protect outgoing messages.

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 using that members's public key. This occurs when verifying countersignatures of received messages (in signature mode), or when verifying messages integrity-protected with pairwise keying material derived from asymmetric keys (in pairwise mode). As a consequence,

group members may end up retaining stale Recipient Contexts, that are no longer useful to verify incoming secure messages.

In order to alleviate this issue, the Group Manager SHOULD NOT recycle 'kid' values within a same group, especially in the short term. Furthermore, applications may define policies to: i) delete (long-)unused Recipient Contexts and reduce the impact on storage space; as well as ii) check with the Group Manager that an owned public key is currently the one associated to a 'kid' value, after a number of consecutive failed verifications.

The distribution of a new Gid and Master Secret may result in temporarily misaligned Security Contexts among group members. In particular, this may result in a group member not able to process messages received right after a new Gid and Master Secret have been distributed. A discussion on practical consequences and possible ways to address them is provided in Section 10.4.

If required by the application (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].

2.5. Exhaustion of Partial IV Values

An endpoint can eventually exhaust its own Sender Sequence Number, which is incremented after sending each new message including a Partial IV. This is the case for all group requests, all Observe notifications [RFC7641] and, optionally, any other response.

If an implementation's integers only support wrapping addition, the implementation MUST detect a wrap-around of the Sender Sequence Number value and treat that as exhausted instead.

Upon exhausting its own Sender Sequence Number, the endpoint MUST NOT transmit further messages for that group 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:

- The Group Manager renews the Security Context in the group (see Section 2.4).

- The Group Manager provides a new Sender ID value to the endpoint that has experienced the exhaustion. Then, the endpoint derives a new Sender Context using the new Sender ID, as described in Section 3.2 of [RFC8613].

In either case, same considerations from Section 2.4 hold about possible retaining of stale Recipient Contexts.

3. Pairwise Keys

Certain signature schemes, such as EdDSA and ECDSA, support a secure combined signature and encryption scheme. This section specifies the derivation of pairwise encryption keys for use in the pairwise and optimized modes of Group OSCORE.

3.1. Key Derivation

Two group members can derive a symmetric pairwise key, from their Sender/Recipient Key and a static-static Diffe-Hellman shared secret [NIST-800-56A]. The key derivation is as follows, and uses the same construction used in Section 3.2.1 of [RFC8613].

Pairwise key = HKDF(Sender/Recipient Key, Shared Secret, info, L)

where:

- The Sender/Recipient key is the Sender Key of the sender, i.e. the Recipient Key that the recipient stores in its own Recipient Context corresponding to the sender.

- The Shared Secret is computed as a static-static Diffie-Hellman shared secret [NIST-800-56A], where the sender uses its own private key and the recipient's public key, while the recipient uses its own private key and the senders's public key. The Shared Secret may be stored in memory, rather than recomputed each time it is needed.

- info and L are defined as in Section 3.2.1 of [RFC8613].

The security of using the same key pair for Diffie-Hellman and for signing is proven in [Degabriele]. The derivation of pairwise keys

defined above is compatible with ECDSA and EdDSA asymmetric keys, but is not compatible with RSA asymmetric keys.

If EdDSA asymmetric keys are used, the Edward coordinates are mapped to Montgomery coordinates using the maps defined in Sections 4.1 and 4.2 of [RFC7748], before using the X25519 and X448 functions defined in Section 5 of [RFC7748].

After completing the establishment of a new Security Context (see Section 2.4), every group member MUST delete all its pairwise keys. Since new Sender/Recipient keys are derived from the new group keying material (see Section 2.2), every group member MUST use such new Sender/Recipient keys when possibly deriving new pairwise keys.

As long as any two group members preserve the same asymmetric keys, the Diffie-Hellman shared secret does not change across updates of the group keying material.

3.2. Usage of Sequence Numbers

When using any of its pairwise keys, a sender endpoint MUST use the current fresh value of its own Sender Sequence Number, from its own Sender Context (see Section 2.2). That is, the same Sender Sequence Number space is used for all outgoing messages sent to the group and protected with Group OSCORE, thus limiting both storage and complexity.

On the other hand, when combining one-to-many and one-to-one communication in the group, this may result in the Partial IV values moving forward more often. Fundamentally, this is due to the fact that not all the recipients receive all messages from a given sender. For instance, requests sent over multicast (in signature mode) are addressed to the whole group, while requests sent over unicast (in signature mode or pairwise mode) are not.

As a consequence, replay checks may be invoked more often on the recipient side, where larger replay windows should be considered.

3.3. Note on Implementation

In order to optimize performance, an endpoint A may derive a pairwise key used with an endpoint B in the OSCORE group only once, and then store it in its own Security Context for future retrieval. This can work as follows.

Endpoint A can have a Pairwise Sender Context associated to B, within its own Sender Context. This Pairwise Sender Context includes:

- The Recipient ID of B for A, i.e. the Sender ID of B.

- The Pairwise Key derived as defined in Section 3, with A acting as sender and B acting as recipient.

More generally, A has one of such Paiwise Sender Contexts within its own Sender Context, for each different intended recipient.

Furthermore, A can additionally store in its own Recipient Context associated to B the Pairwise Key to use for incoming traffic from B. That is, this Pairwise Key is derived as defined in Section 3, with A acting as recipient and B acting as sender.

4. The COSE Object

Building on Section 5 of [RFC8613], 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 Associated Data (AEAD) algorithm. For the signature mode of Group OSCORE the following modifications apply.

4.1. Counter Signature

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 as described in Appendix A.2 of [RFC8152], by using its own private key and according to the Counter Signature Algorithm and Counter Signature Parameters in the Security Context. In particular, the Sig_structure contains the external_aad as defined in Section 4.3.2 and the ciphertext of the COSE_Encrypt0 object as payload.

4.2. The 'kid' and 'kid context' parameters

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 [RFC8613], the 'kid' parameter is always present in all messages, i.e. both requests and responses.

The value of the 'kid context' parameter in the 'unprotected' field of requests messages MUST be set to the ID Context, i.e. the Group Identifier value (Gid) of the group's Security Context. That is, unlike in [RFC8613], the 'kid context' parameter is always present in requests.

4.3. external_aad

The external_aad of the Additional Authenticated Data (AAD) is built differently. In particular, it has one structure used for the encryption process producing the ciphertext, and a second structure used for the signing process producing the counter signature.

4.3.1. external_aad for Encryption

The first external_aad structure used for the encryption process producing the ciphertext (see Section 5.3 of [RFC8152]) includes also the counter signature algorithm and related parameters used to sign messages. In particular, compared with Section 5.4 of [RFC8613], the 'algorithms' array in the aad_array MUST also include:

- '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 11.1) for this counter signature algorithm.

- '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 11.1), for the used counter signature algorithm. If the Counter Signature Parameters in the Common Context is empty, 'par_countersign' MUST be encoding the CBOR simple value Null.

- 'par_countersign_key', which contains the Counter Signature Key Parameters from the Common Context (see Section 2). This parameter contains the counter signature key parameters encoded as specified in the "Parameters" field of the Counter Signature Key Parameters Registry (see Section 11.2), for the used counter signature algorithm. If the Counter Signature Key Parameters in the Common Context is empty, 'par_countersign_key' MUST be encoding the CBOR simple value Null.

Thus, the following external_aad structure is used for the encryption process producing the ciphertext (see Section 5.3 of [RFC8152]).

```
external_aad = bstr .cbor aad_array

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

4.3.2. external_aad for Signing

The second external_aad structure used for the signing process producing the counter signature as defined below includes also:

- the counter signature algorithm and related parameters used to sign messages, encoded as in the external_aad structure defined in Section 4.3.1;

- the value of the OSCORE Option included in the OSCORE message, encoded as a binary string.

Thus, the following external_aad structure is used 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 / nil,
                 par_countersign_key : any / nil],
   request_kid : bstr,
   request_piv : bstr,
   options : bstr,
   OSCORE_option: bstr
]
```

Note for implementation: this requires the value of the OSCORE option to be fully ready, before starting the signing process. Also, this requires that the aad_array is long enough to contain the longest possible OSCORE option.

5. OSCORE Header Compression

The OSCORE header compression defined in Section 6 of [RFC8613] is used, with the following differences.

- The payload of the OSCORE message SHALL encode the ciphertext of the COSE object. In the signature mode as well as in the optimized compressed requests of the optimized mode (see Section 9.1.1), the ciphertext above is concatenated with the value of the CounterSignature0 of the COSE object, computed as described in Section 4.1.

- This specification defines the usage of the sixth least significant bit, namely the Pairwise Flag bit, in the first byte of the OSCORE option containing the OSCORE flag bits. This flag bit is registered in Section 11.3 of this specification.

- The Pairwise Flag bit is set to 1 if the OSCORE message is protected using pairwise keying material, which is shared with a single group member as single intended recipient and derived as defined in Section 3. This is used, for instance, to send responses with the optimized mode defined in Section 9. In any other case, especially when the OSCORE message is protected as per Section 7.1 and Section 7.3, this bit MUST be set to 0.

  If any of the following conditions holds, a recipient MUST discard an incoming OSCORE message with the Pairwise Flag bit set to 1:

  * The recipient does not support this feature, i.e. it is not capable or willing to process OSCORE messages protected using pairwise keying material.

  * The recipient can not retrieve a Security Context which is both valid to process the message and also associated to an OSCORE group.

5.1. Examples of Compressed COSE Objects

This section covers a list of OSCORE Header Compression examples for group requests and responses, with Group OSCORE used in signature mode.

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 5 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 [RFC8613] has value 10. The examples also assume that the plaintext (see Section 5.3 of [RFC8613]) is 6 bytes long, and that the AEAD tag is 8 bytes long, hence resulting in a ciphertext which is 14 bytes long. Finally, COUNTERSIGN is the CounterSignature0 byte string as described in Section 4 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 = 0x60b035059d9ef5667c5a0710823b, 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)

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

The requirements and properties described in Section 7 of [RFC8613] 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.

6.1. 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.

7. Message Processing

Each request message and response message is protected and processed as specified in [RFC8613], with the modifications described in the following sections. In particular, the following sections refer to the signature mode of Group OSCORE, while the optimized mode and the pairwise mode are described in Section 9 and Appendix G, respectively.

The following security objectives are fulfilled, as further discussed in Appendix A.2: data replay protection, group-level data confidentiality, source authentication and message integrity.

As per [RFC7252][I-D.ietf-core-groupcomm-bis], group requests sent over multicast MUST be Non-Confirmable, and thus cannot be retransmitted by the CoAP messaging layer. Instead, applications should store such outgoing messages for a pre-defined, sufficient amount of time, in order to correctly perform possible retransmissions at the application layer. 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 [RFC8613]. However, a recipient MUST stop processing and silently reject any message which is malformed and does not follow the format specified in Section 4, or which is not cryptographically validated in a successful way. In 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.

7.1. Protecting the Request

A client transmits a secure group request as described in Section 8.1 of [RFC8613], with the following modifications.

- In step 2, the 'algorithms' array in the Additional Authenticated Data is modified as described in Section 4 of this specification.

- In step 4, the encryption of the COSE object is modified as described in Section 4 of this specification. The encoding of the compressed COSE object is modified as described in Section 5 of this specification.

- In step 5, the counter signature is computed and the format of the OSCORE message is modified as described in Section 4 and Section 5 of this specification. In particular, the payload of the OSCORE message includes also the counter signature.

7.1.1. Supporting Observe

If Observe [RFC7641] is supported, for each newly started observation, the client MUST store the value of the 'kid' parameter from the original Observe request.

The client MUST NOT update the stored value, even in case it is individually rekeyed and receives a new Sender ID from the Group Manager (see Section 2.5).

7.2. Verifying the Request

Upon receiving a secure group request, a server proceeds as described in Section 8.2 of [RFC8613], with the following modifications.

- In step 2, the decoding of the compressed COSE object follows Section 5 of this specification. In particular:
  - If the Pairwise Flag bit is set to 1, and the server discards the request due to not supporting this feature or not retrieving a Security Context associated to the OSCORE group, the server MAY respond with a 4.02 (Bad Option) error. When doing so, the server MAY set an Outer Max-Age option with value zero, and MAY include a descriptive string as diagnostic payload.
  - If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the server MAY create a new Recipient Context and initializes it according to Section 3 of [RFC8613], also retrieving the client's public key. Such a configuration is application specific. If the application does not specify dynamic derivation of new Recipient Contexts, then the server SHALL stop processing the request.
- In step 4, the 'algorithms' array in the Additional Authenticated Data is modified as described in Section 4 of this specification.
- In step 6, the server also verifies the counter signature using the public key of the client from the associated Recipient Context. If the signature verification fails, the server MAY reply with a 4.00 (Bad Request) response.
- 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.

A server SHOULD NOT process a request if the received Recipient ID ('kid') is equal to its own Sender ID in its own Sender Context.

7.2.1. Supporting Observe

If Observe [RFC7641] is supported, for each newly started observation, the server MUST store the value of the 'kid' parameter from the original Observe request.

The server MUST NOT update the stored value, even in case the observer client is individually rekeyed and starts using a new Sender ID received from the Group Manager (see Section 2.5).

7.3. 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 [RFC8613], with the following modifications.

- In step 2, the 'algorithms' array in the Additional Authenticated Data is modified as described in Section 4 of this specification.

- In step 4, the encryption of the COSE object is modified as described in Section 4 of this specification. The encoding of the compressed COSE object is modified as described in Section 5 of this specification.

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

Note that the server MUST always protect a response by using its own Sender Context from the latest owned Security Context.

Consistently, upon the establishment of a new Security Context, the server may end up protecting a response by using a Security Context different from the one used to protect the group request (see Section 10.4). In such a case:

- The server MUST encode the Partial IV (Sender Sequence Number in network byte order), which is set to the Sender Sequence Number of the server; increment the Sender Sequence Number by one; compute the AEAD nonce from the Sender ID, Common IV, and Partial IV.

- The server MUST include in the response the 'Partial IV' parameter, which is set to the encoded Partial IV value above.

- The server SHOULD include in the response the 'kid context' parameter, which is set to the ID Context of the new Security Context, i.e. the new Group Identifier (Gid).

7.3.1. Supporting Observe

If Observe [RFC7641] is supported, the server may have ongoing observations, started by Observe requests protected with an old Security Context.

After completing the establishment of a new Security Context, the server MUST protect the following notifications with its own Sender Context from the new Security Context.

For each ongoing observation, the server SHOULD include in the first notification protected with the new Security Context also the 'kid context' parameter, which is set to the ID Context of the new Security Context, i.e. the new Group Identifier (Gid). It is OPTIONAL for the server to include the 'kid context' parameter, as set to the new Gid, also in further following notifications for those observations.

Furthermore, for each ongoing observation, the server MUST use the stored value of the 'kid' parameter from the original Observe request, as value for the 'request_kid' parameter in the two external_aad structures (see Section 4.3.1 and Section 4.3.2), when protecting notifications for that observation.

7.4. Verifying the Response

Upon receiving a secure response message, the client proceeds as described in Section 8.4 of [RFC8613], with the following modifications.

- In step 2, the decoding of the compressed COSE object is modified as described in Section 4 of this specification. If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the client MAY create a new Recipient Context and initializes it according to Section 3 of [RFC8613], also retrieving the server's public key. If the application does not specify dynamic derivation of new Recipient Contexts, then the client SHALL stop processing the response.

- In step 3, the 'algorithms' array in the Additional Authenticated Data is modified as described in Section 4 of this specification.

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

- 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.

Note that, as discussed in Section 10.4, a client may receive a response protected with a Security Context different from the one used to protect the corresponding group request.

7.4.1. Supporting Observe

If Observe [RFC7641] is supported, for each ongoing observation, the client MUST use the stored value of the 'kid' parameter from the original Observe request, as value for the 'request_kid' parameter in the two external_aad structures (see Section 4.3.1 and Section 4.3.2), when verifying notifications for that observation.

This ensures that the client can correctly verify notifications, even in case it is individually rekeyed and starts using a new Sender ID received from the Group Manager (see Section 2.5).

8. 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.

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

4. Establishing the Common Context part of the Security Context, and providing it to authorized group members during the join process, together with the corresponding 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.

11. Validating that the format and parameters of public keys of group members are consistent with the countersignature algorithm and related parameters used in the respective OSCORE group.

9. Optimized Mode

For use cases that do not require an intermediary performing signature verification and that use a compatible signature algorithm, the optimized mode defined in this section provides significant smaller message sizes and increases the security by making responses confidential to other group members than the intended recipient.

9.1. Optimized Request

No changes.

9.1.1. Optimized Compressed Request

The OSCORE header compression defined in Section 5 is used, with the following difference: the payload of the OSCORE message SHALL encode the ciphertext without the tag, concatenated with the value of the CounterSignature0 of the COSE object computed as described in Section 4.1.

The optimized compressed request is compatible with all AEAD algorithms defined in [RFC8152], but would not be compatible with AEAD algorithms that do not have a well-defined tag.

9.2. Optimized Response

An optimized response is protected as defined in Section 7.3, with the following differences.

- The server MUST set to 1 the sixth least significant bit of the OSCORE flag bits in the OSCORE option, i.e. the Pairwise Flag.

- The COSE_Encrypt0 object included in the optimized response is encrypted using a symmetric pairwise key K, that the server derives as defined in Section 3. In particular, the Sender/ Recipient Key is the Sender Key of the server from its own Sender Context, i.e. the Recipient Key that the client stores in its own Recipient Context corresponding to the server.

- The Counter Signature is not computed. That is, unlike defined in Section 5, the payload of the OSCORE message terminates with the encoded ciphertext of the COSE object.

Note that no changes are made to the AEAD nonce and AAD.

Upon receiving a response with the Pairwise Flag set to 1, the client MUST process it as defined in Section 7.4, with the following differences.

- No countersignature to verify is included.

- The COSE_Encrypt0 object included in the optimized response is decrypted and verified using the same symmetric pairwise key K, that the client derives as described above for the server side and as defined in Section 3.

9.2.1. Optimized Compressed Response

No changes.

10. Security Considerations

The same threat model discussed for OSCORE in Appendix D.1 of [RFC8613] holds for Group OSCORE. In addition, source authentication of messages is explicitly ensured by means of counter signatures, as discussed in Section 10.1.

The same considerations on supporting Proxy operations discussed for OSCORE in Appendix D.2 of [RFC8613] hold for Group OSCORE.

The same considerations on protected message fields for OSCORE discussed in Appendix D.3 of [RFC8613] hold for Group OSCORE.

The same considerations on uniqueness of (key, nonce) pairs for OSCORE discussed in Appendix D.4 of [RFC8613] hold for Group OSCORE. This is further discussed in Section 10.2.

The same considerations on unprotected message fields for OSCORE discussed in Appendix D.5 of [RFC8613] hold for Group OSCORE, with the following difference. The countersignature included in a Group OSCORE message is computed also over the value of the OSCORE option, which is part of the Additional Authenticated Data used in the signing process. This is further discussed in Section 10.6.

As discussed in Section 6.2.3 of [I-D.ietf-core-groupcomm-bis], Group OSCORE addresses security attacks against CoAP listed in Sections 11.2-11.6 of [RFC7252], especially when mounted over IP multicast.

The rest of this section first discusses security aspects to be taken into account when using Group OSCORE. Then it goes through aspects covered in the security considerations of OSCORE (Section 12 of [RFC8613]), and discusses how they hold when Group OSCORE is used.

10.1. Group-level Security

The signature mode described in Section 7 relies on commonly shared group keying material to protect communication within a group. This has the following implications.

- 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.

- 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.

Instead, the pairwise mode described in Appendix G protects messages by using pairwise symmetric keys, derived from the static-static Diffie-Hellman shared secret computed from the asymmetric keys of the sender and recipient endpoint (see Section 3). Therefore, in the parwise mode, the AEAD algorithm provides both pairwise data-confidentiality and source authentication of messages, without using counter signatures.

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.

10.2. Uniqueness of (key, nonce)

The proof for uniqueness of (key, nonce) pairs in Appendix D.4 of [RFC8613] is also valid in group communication scenarios. That is, given an OSCORE group:

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

- The case A in Appendix D.4 of [RFC8613] concerns all group requests and responses including a Partial IV (e.g. Observe notifications). In this case, same considerations from [RFC8613] apply here as well.

- The case B in Appendix D.4 of [RFC8613] concerns responses not including a Partial IV (e.g. single response to a group request). In this case, same considerations from [RFC8613] 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.

10.3. 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.

10.4. 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.

This may result in a client using an old Security Context to protect a group request, and a server using a different new Security Context to protect a corresponding response. That is, clients may receive a response protected with a Security Context different from the one used to protect the corresponding group request.

In particular, a server may first get a group request protected with the old Security Context, then install the new Security Context, and only after that produce a response to send back to the client. Since a sender always protects an outgoing message using the latest owned Security Context, the server discussed above protects the possible response using the new Security Context. Then, the client will process that response using the new Security Context, provided that it has installed the new security parameters and keying material before the message reception.

In case block-wise transfer [RFC7959] is used, the same considerations from Section 7.2 of [I-D.ietf-ace-key-groupcomm] hold.

Furthermore, as described below, a group rekeying may temporarily result in misaligned Security Contexts between the sender and recipient of a same message.

10.4.1. Late Update on the Sender

In this 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. This makes particular sense when the recipient is a client, that would hence be able to process incoming responses protected with the old, recent, Security Context used to protect the associated group request. Instead, a recipient server would better and more simply discard an incoming group request which is not successfully processed with the new Security Context.

This tolerance preserves the processing of secure messages throughout a long-lasting key rotation, as group rekeying processes may likely take a long time to complete, especially in large scale groups. On the other hand, a former (compromised) group member can abusively take advantage of this, and send messages protected with the old retained Security Context. Therefore, a conservative application policy should not admit the retention of old Security Contexts.

10.4.2. Late Update on the Recipient

In this case, the sender protects a message using the new Security Context, but the recipient receives that message before having installed the new Security Context. Therefore, the recipient would not be able to correctly process the message and hence discards it.

If the recipient installs 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.

10.5. 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.

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, AEAD keys are different among different groups.

The entity assigning an IP multicast address may help limiting the chances to experience such collisions of Group Identifiers. In particular, it may allow the Group Managers of groups using the same IP multicast address to share their respective list of assigned Group Identifiers currently in use.

10.6. Cross-group Message Injection

A same endpoint is allowed to and would likely use the same signature key in multiple OSCORE groups, possibly administered by different Group Managers. Also, the same endpoint can register several times in the same group, getting multiple unique Sender IDs. This requires that, when a sender endpoint sends a message to an OSCORE group using

a Sender ID, the countersignature included in the message is explicitly bound also to that group and to the used Sender ID.

To this end, the countersignature of each message protected with Group OSCORE is computed also over the value of the OSCORE option, which is part of the Additional Authenticated Data used in the signing process (see Section 4.3.2). That is, the countersignature is computed also over: the ID Context (Group ID) and the Partial IV, which are always present in group requests; as well as the Sender ID of the message originator, which is always present in all group requests and responses.

Since the signing process takes as input also the ciphertext of the COSE_Encrypt0 object, the countersignature is bound not only to the intended OSCORE group, hence to the triplet (Master Secret, Master Salt, ID Context), but also to a specific Sender ID in that group and to its specific symmetric key used for AEAD encryption, hence to the quartet (Master Secret, Master Salt, ID Context, Sender ID).

This makes it practically infeasible to perform the attack described below, where a malicious group member injects forged messages to a different OSCORE group than the originally intended one. Let us consider:

- Two OSCORE groups G1 and G2, with ID Context (Group ID) Gid1 and Gid2, respectively. Both G1 and G2 use the AEAD cipher AES-CCM-16-64-128, i.e. the MAC of the ciphertext is 8 bytes in size.

- A victim endpoint V which is member of both G1 and G2, and uses the same signature key in both groups. The endpoint V has Sender ID Sid1 in G1 and Sender ID Sid2 in G2. The pairs (Sid1, Gid1) and (Sid2, Gid2) identify the same public key of V in G1 and G2, respectively.

- A malicious endpoint Z is also member of both G1 and G2. Hence, Z is able to derive the symmetric keys associated to V in G1 and G2.

If countersignatures were not computed also over the value of the OSCORE option as discussed above, Z can intercept a group message M1 sent by V to G1, and forge a valid signed message M2 to be injected in G2, making it appear as sent by V and valid to be accepted.

More in detail, Z first intercepts a message M1 sent by V in G1, and tries to forge a message M2, by changing the value of the OSCORE option from M1 as follows: the 'kid context' is changed from G1 to G2; and the 'kid' is changed from Sid1 to Sid2.

If M2 is used as a request message, there is a probability equal to 2^-64 that the same unchanged MAC is successfully verified by using Sid2 as 'request_kid' and the symmetric key associated to V in G2. In such a case, the same unchanged signature would be also valid. Note that Z can check offline if a performed forgery is actually valid before sending the forged message to G2. That is, this attack has a complexity of 2^64 offline calculations.

If M2 is used as a response, Z can also change the response Partial IV, until the same unchanged MAC is successfully verified by using Sid2 as 'request_kid' and the symmetric key associated to V in G2. In such a case, the same unchanged signature would be also valid. Since the Partial IV is 5 bytes in size, this requires 2^40 operations to test all the Partial IVs, which can be done in real-time. Also, the probability that a single given message M1 can be used to forge a response M2 for a given request is equal to 2^-24, since there are more MAC values (8 bytes in size) than Partial IV values (5 bytes in size).

Note that, by changing the Partial IV as discussed above, any member of G1 would also be able to forge a valid signed response message M2 to be injected in G1.

10.7. Group OSCORE for Unicast Requests

With reference to the processing defined in Section 7.1 for the signature mode and in Section 9.1.1 for the optimized mode, it is NOT RECOMMENDED for a client to use Group OSCORE for securing a request sent to a single group member over unicast.

If the client uses its own Sender Key to protect a unicast request to a group member, an on-path adversary can, right then or later on, redirect that request to one/many different group member(s) over unicast, or to the whole OSCORE group over multicast. By doing so, the adversary can induce the target group member(s) to perform actions intended to one group member only. Note that the adversary can be external, i.e. (s)he does not need to also be a member of the OSCORE group.

This is due to the fact that the client is not able to indicate the single intended recipient in a way which is secure and possible to process for Group OSCORE on the server side. In particular, Group OSCORE does not protect network addressing information such as the IP address of the intended recipient server. It follows that the server(s) receiving the redirected request cannot assert whether that was the original intention of the client, and would thus simply assume so.

With particular reference to block-wise transfers [RFC7959], Section 2.3.6 of [I-D.ietf-core-groupcomm-bis] points out that, while an initial request including the CoAP Block2 option can be sent over multicast, any other request in a transfer has to occur over unicast, individually addressing the servers in the group.

Additional considerations are discussed in Appendix E.3, with respect to requests including an Echo Option [I-D.ietf-core-echo-request-tag] that has to be sent over unicast, as a challenge-response method for servers to achieve synchronization of client Sender Sequence Numbers.

The impact of such an attack depends especially on the REST method of the request, i.e. the Inner CoAP Code of the OSCORE request message. In particular, safe methods such as GET and FETCH would trigger (several) unintended responses from the targeted server(s), while not resulting in destructive behavior. On the other hand, non safe methods such as PUT, POST and PATCH/iPATCH would result in the target server(s) taking active actions on their resources and possible cyber-physical environment, with the risk of destructive consequences and possible implications for safety.

A client may instead use the pairwise mode defined in Appendix G.2, in order to protect a request sent to a single group member by using pairwise keying material (see Section 3). This prevents the attack discussed above by construction, as only the intended server is able to derive the pairwise keying material used by the client to protect the request. A client supporting the pairwise mode SHOULD use it to protect requests sent to a single group member over unicast, instead of using the signature mode.

10.8. End-to-end Protection

The same considerations from Section 12.1 of [RFC8613] hold for Group OSCORE.

Additionally, (D)TLS and Group OSCORE can be combined for protecting message exchanges occurring over unicast. Instead, it is not possible to combine DTLS and Group OSCORE for protecting message exchanges where messages are (also) sent over multicast.

10.9. Security Context Establishment

The use of COSE_Encrypt0 and AEAD to protect messages as specified in this document requires an endpoint to be a member of an OSCORE group.

That is, upon joining the group, the endpoint securely receives from the Group Manager the necessary input parameters, which are used to

derive the Common Context and the Sender Context (see Section 2). The Group Manager ensures uniqueness of Sender IDs in the same group.

Each different Recipient Context for decrypting messages from a particular sender can be derived at runtime, at the latest upon receiving a message from that sender for the first time.

Countersignatures of group messages are verified by means of the public key of the respective sender endpoint. Upon nodes' joining, the Group Manager collects such public keys and MUST verify proof-of-possession of the respective private key. Later on, a group member can request from the Group Manager the public keys of other group members.

The joining process can occur, for instance, as defined in [I-D.ietf-ace-key-groupcomm-oscore].

10.10. Master Secret

Group OSCORE derives the Security Context using the same construction as OSCORE, and by using the Group Identifier of a group as the related ID Context. Hence, the same required properties of the Security Context parameters discussed in Section 3.3 of [RFC8613] hold for this document.

With particular reference to the OSCORE Master Secret, it has to be kept secret among the members of the respective OSCORE group and the Group Manager responsible for that group. Also, the Master Secret must have a good amount of randomness, and the Group Manager can generate it offline using a good random number generator. This includes the case where the Group Manager rekeys the group by generating and distributing a new Master Secret. Randomness requirements for security are described in [RFC4086].

10.11. Replay Protection

As in OSCORE, also Group OSCORE relies on sender sequence numbers included in the COSE message field 'Partial IV' and used to build AEAD nonces.

Note that the Partial IV of an endpoint does not necessarily grow monotonically. For instance, upon exhaustion of the endpoint Sender Sequence Number, the Partial IV also gets exhausted. As discussed in Section 2.5, this results either in the endpoint being individually rekeyed and getting a new Sender ID, or in the establishment of a new Security Context in the group. Therefore, uniqueness of (key, nonce) pairs (see Section 10.2) is preserved also when a new Security Context is established.

As discussed in Section 6.1, an endpoint that has just joined a group is exposed to replay attack, as it is not aware of the sender sequence numbers currently used by other group members. Appendix E describes how endpoints can synchronize with senders' sequence numbers.

Unless exchanges in a group rely only on unicast messages, Group OSCORE cannot be used with reliable transport. Thus, unless only unicast messages are sent in the group, it cannot be defined that only messages with sequence numbers that are equal to the previous sequence number + 1 are accepted.

The processing of response messages described in Section 7.4 also ensures that a client accepts a single valid response to a given request from each replying server, unless CoAP observation is used.

10.12. Client Aliveness

As discussed in Section 12.5 of [RFC8613], a server may use the Echo option [I-D.ietf-core-echo-request-tag] to verify the aliveness of the client that originated a received request. This would also allow the server to (re-)synchronize with the client's sequence number, as well as to ensure that the request is fresh and has not been replayed or (purposely) delayed, if it is the first one received from that client after having joined the group or rebooted (see Appendix E.3).

10.13. Cryptographic Considerations

The same considerations from Section 12.6 of [RFC8613] about the maximum Sender Sequence Number hold for Group OSCORE.

As discussed in Section 2.5, an endpoint that experiences a exhaustion of its own Sender Sequence Number MUST NOT transmit further messages including a Partial IV, until it has derived a new Sender Context. This prevents the endpoint to reuse the same AEAD nonces with the same Sender key.

In order to renew its own Sender Context, the endpoint SHOULD inform the Group Manager, which can either renew the whole Security Context by means of group rekeying, or provide only that endpoint with a new Sender ID value. In either case, the endpoint derives a new Sender Context, and in particular a new Sender Key.

Additionally, the same considerations from Section 12.6 of [RFC8613] hold for Group OSCORE, about building the AEAD nonce and the secrecy of the Security Context parameters.

The EdDSA signature algorithm Ed25519 [RFC8032] is mandatory to implement. For many constrained IoT devices, it is problematic to support more than one signature algorithm or multiple whole cipher suites. This means that some deployments using, for instance, ECDSA with NIST P-256 may not support the mandatory signature algorithm. However, this is not a problem for local deployments.

10.14. Message Segmentation

The same considerations from Section 12.7 of [RFC8613] hold for Group OSCORE.

10.15. Privacy Considerations

Group OSCORE ensures end-to-end integrity protection and encryption of the message payload and all options that are not used for proxy operations. In particular, options are processed according to the same class U/I/E that they have for OSCORE. Therefore, the same privacy considerations from Section 12.8 of [RFC8613] hold for Group OSCORE.

Furthermore, the following privacy considerations hold, about the OSCORE option that may reveal information on the communicating endpoints.

- The 'kid' parameter, which is intended to help a recipient endpoint to find the right Recipient Context, may reveal information about the Sender Endpoint. Since both requests and responses always include the 'kid' parameter, this may reveal information about both a client sending a group request and all the possibly replying servers sending their own individual response.

- The 'kid context' parameter, which is intended to help a recipient endpoint to find the right Recipient Context, reveals information about the sender endpoint. In particular, it reveals that the sender endpoint is a member of a particular OSCORE group, whose current Group ID is indicated in the 'kid context' parameter. Moreover, this parameter explicitly relates two or more communicating endpoints, as members of the same OSCORE group.

Also, using the mechanisms described in Appendix E.3 to achieve sequence number synchronization with a client may reveal when a server device goes through a reboot. This can be mitigated by the server device storing the precise state of the replay window of each known client on a clean shutdown.

Finally, the mechanism described in Section 10.5 to prevent collisions of Group Identifiers from different Group Managers may reveal information about events in the respective OSCORE groups. In particular, a Group Idenfier changes when the corresponding group is rekeyed. Thus, changes in the shared list of Group Identifiers may be used to infer about the rate and patterns of group membership changes triggering a group rekeying, e.g. due to newly joined members or evicted (compromised) members. In order to alleviate such privacy concerns, it should be hidden from the Group Managers which exact Group Manager has currently assigned which Group Identifiers in its OSCORE groups.

11. 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.

11.1. 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 11.4.

This registry specifies the parameters of each admitted countersignature algorithm, as well as the possible structure they are organized into. This information is used to populate the parameter Counter Signature Parameters of the Common Context (see Section 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.

- 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.

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

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

- 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]

11.2. Counter Signature Key Parameters Registry

This specification establishes the IANA "Counter Signature Key Parameters" Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.4.

This registry specifies the parameters of countersignature keys for each admitted countersignature algorithm, as well as the possible structure they are organized into. This information is used to populate the parameter Counter Signature Key Parameters of the Common Context (see Section 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.

- 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.

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

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

- 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	[kty : int , crv : int]	kty value is 1, as Key Type "OKP" from the COSE Key Types Registry crv value taken from the COSE	[This Document]

			Elliptic Curve Registry		
ES256	-7	[kty : int , crv : int]	kty value is 2, as Key Type "EC2" from the COSE Key Types Registry crv value taken from the COSE Elliptic Curve Registry	[This Document]	
ES384	-35	[kty : int , crv : int]	kty value is 2, as Key Type "EC2" from the COSE Key Types Registry crv value taken from the COSE Elliptic Curve Registry	[This Document]	
ES512	-36	[kty : int , crv : int]	kty value is 2, as Key Type "EC2" from the COSE Key Types Registry crv value taken from the COSE Elliptic Curve Registry	[This Document]	
PS256	-37	kty : int	kty value is 3, as Key Type "RSA" from the COSE Key Types Registry	[This Document]	

PS384	-38	kty : int	kty value is 3, as Key Type "RSA" from the COSE Key Types Registry	[This Document]	
PS512	-39	kty : int	kty value is 3, as Key Type "RSA" from the COSE Key Types Registry	[This Document]	

11.3. OSCORE Flag Bits Registry

IANA is asked to add the following value entry to the "OSCORE Flag Bits" subregistry defined in Section 13.7 of [RFC8613] as part of the "CoRE Parameters" registry.

Bit Position	Name	Description	Reference
2	Pairwise Protection Flag	Set to 1 if the message is protected with pairwise keying material	[This Document]

11.4. Expert Review Instructions

The IANA Registries established in this document are 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:

- Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the requested entries. Experts should inspect the entry for the algorithm considered, to verify the conformity of the encoding proposed against the theoretical algorithm, including completeness of the 'Parameters' column. Expert needs to make sure values are taken from the right registry, when that's required. Expert should consider requesting an opinion on the correctness of registered parameters from the CBOR Object Signing and Encryption Working Group (COSE).

Encodings that do not meet these objective of clarity and completeness should not be registered.

- Duplicated registration and point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.

- Experts should take into account the expected usage of fields when approving point assignment. The length of the 'Parameters' encoding should be weighed against the usage of the entry, considering the size of device it will be used on. Additionally, the length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

- Specifications are recommended. When specifications are not provided, the description provided needs to have sufficient information to verify the points above.

12. References

12.1. Normative References

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", draft-ietf-core-groupcomm-bis-00 (work in progress), March 2020.

[NIST-800-56A]
Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography - NIST Special Publication 800-56A, Revision 3", April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[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>.

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

12.2. Informative References

[Degabriele]
Degabriele, J., Lehmann, A., Paterson, K., Smart, N., and M. Strefler, "On the Joint Security of Encryption and Signature in EMV", December 2011, <https://eprint.iacr.org/2011/615>.

[I-D.ietf-ace-key-groupcomm]
Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", draft-ietf-ace-key-groupcomm-05 (work in progress), March 2020.

[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-05 (work in progress), March 2020.

[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-33 (work in progress), February 2020.

[I-D.ietf-core-echo-request-tag]
Amsuess, C., Mattsson, J., and G. Selander, "CoAP: Echo, Request-Tag, and Token Processing", draft-ietf-core-echo-request-tag-09 (work in progress), March 2020.

[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>.

[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>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

Appendix A. Assumptions and Security Objectives

This section presents a set of assumptions and security objectives for the approach described in this document. The rest of this section refers to three types of groups:

- Application group, i.e. a set of CoAP endpoints that share a common pool of resources.

- Security group, as defined in Section 1.1 of this specification. There can be a one-to-one or a one-to-many relation between security groups and application groups. Any two application groups associated to the same security group do not share any same resource.

- CoAP group, as defined in [I-D.ietf-core-groupcomm-bis] i.e. a set of CoAP endpoints, where each endpoint is configured to receive CoAP multicast requests that are sent to the group's associated IP multicast address and UDP port. An endpoint may be a member of multiple CoAP groups. There can be a one-to-one or a one-to-many relation between CoAP groups and application groups. Note that a device sending a CoAP request to a CoAP group is not necessarily itself a member of that group: it is a member only if it also has a CoAP server endpoint listening to requests for this CoAP group, sent to the associated IP multicast address and port. In order to provide secure group communication, all members of a CoAP group as well as all further endpoints configured only as clients sending CoAP (multicast) requests to the CoAP group have to be member of a security group.

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 CoAP 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 clients transmit data to the CoAP group. According to [I-D.ietf-core-groupcomm-bis], any possible proxy entity is supposed to know about the clients 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 CoAP group.

- Group size: security solutions for group communication should be able to adequately support different and possibly large security groups. The group size is the current number of members in a security 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. Security groups larger than that should be divided into smaller independent groups.

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

- Provisioning and management of Security Contexts: a Security Context must be established among the members of the security group. A secure mechanism must be used to generate, revoke and (re-)distribute keying material, multicast security policies and security parameters in the security group. The actual provisioning and management of the Security Context is out of the scope of this document.

- Multicast data security ciphersuite: all members of a security group 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.

- Backward security: a new device joining the security group should not have access to any old Security Contexts used before its joining. This ensures that a new member of the security group is not able to decrypt confidential data sent before it has joined the security 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 in the security group upon a new member's joining has to be defined as part of the group key management scheme.

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

- Data replay protection: group request messages or response messages replayed within the security group must be detected.

- Group-level data confidentiality: messages sent within the security group shall be encrypted if privacy sensitive data is exchanged within the security 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 security group, but not by an external adversary or other external entities.

- Source authentication: messages sent within the security group shall be authenticated. That is, it is essential to ensure that a message is originated by a member of the security group in the

first place, and in particular by a specific member of the security group.

- Message integrity: messages sent within the security 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 members of the security group.

- Message ordering: it must be possible to determine the ordering of messages coming from a single sender. In accordance with OSCORE [RFC8613], 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 [I-D.ietf-core-groupcomm-bis] 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 [I-D.ietf-core-groupcomm-bis] to understand the non-security related details. This section discusses a number of use cases that benefit from secure group communication, and refers to the three types of groups from Appendix A. 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 lighting devices acting as servers are organized into application groups and CoAP groups, according to their physical location in the building. For instance, lighting devices in a room or corridor can be configured as members of a single application group and corresponding CoAP group. Those ligthing devices together with the switches acting as clients in the same room or corridor can be configured as members of the corresponding security group. Switches are then used to control the lighting devices by sending on/off/dimming commands to all lighting devices in the CoAP 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 with a role in the lighting application 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 set of connected lights, ensuring that the light preset (e.g. dimming level or color) of a large set 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 set 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 set 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 application groups and CoAP 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 application group and corresponding CoAP 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 Low-power and Lossy Network (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 set 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 set 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 Low-power and Lossy Network (LLN) 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 application group and corresponding CoAP 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 latter 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 Multicast Protocol for Low-Power and Lossy Networks (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.

For each group, 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 2.4). 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 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 10.5, 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 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 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.

With the notable exception of Observe notifications and responses following a group rekeying, it is optional for the server to use its own sender sequence number as Partial IV. Instead, for efficiency reasons, the server may rather use the request's Partial IV when protecting a response.

Since it provides no assurance as to freshness of requests, it is thus RECOMMENDED that a server using this synchronization approach always uses its own sender sequence number as Partial IV when protecting a response.

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. 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.

With the notable exception of Observe notifications and responses following a group rekeying, it is optional for the server to use its own sender sequence number as Partial IV. Instead, for efficiency reasons, the server may rather use the request's Partial IV when protecting a response.

In case the baseline synchronization state related to a client is lost, it is RECOMMENDED that the server uses its own sender sequence number as Partial IV when protecting a response to that client, until a new baseline synchronization state for that client is established.

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 Appendix B.1.2 of [RFC8613].

That is, upon receiving a group request from a particular client for the first time, the server processes the message as described in this specification, but, even if valid, does not deliver it to the application. Instead, the server replies to the client with an OSCORE protected 4.01 (Unauthorized) response message, including only the Echo Option and no diagnostic payload. The server stores the option value included therein.

Upon receiving a 4.01 (Unauthorized) 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 4).

Upon receiving the unicast request including the Echo Option, the server verifies that the option value equals the stored and previously sent value. If not, the server MUST NOT process the request further and MAY send a 4.01 (Unauthorized) response including an Echo option.

In case of positive verification, the request is further processed and verified. 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.

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.

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.

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 4.01 (Unauthorized) response to the client. Therefore, silent servers should adopt alternative approaches to achieve and maintain synchronization with sender sequence numbers of clients.

Since requests including the Echo Option are sent over unicast, a server can be a victim of the attack discussed in Section 10.7, when such requests are protected with the signature mode of Group OSCORE, as described in Section 7.1.

Instead, protecting requests with the Echo Option by using the pairwise mode of Group OSCORE as described in Appendix G.2 prevents the attack in Section 10.7. In fact, only the exact server involved in the Echo exchange is able to derive the correct pairwise key used by the client to protect the request including the Echo Option.

In either case, an internal on-path adversary would not be able to mix up the Echo Option value of two different unicast requests, sent by a same client to any two different servers in the group. In fact, this would require the adversary to forge the client's counter signature in both such requests. As a consequence, each of the two servers remains able to selectively accept a request with the Echo Option only if it is waiting for that exact integrity-protected Echo Option value, and is thus the intended recipient.

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. Pairwise Mode

For use cases that do not require an intermediary performing signature verification and that use a compatible signature algorithm, the pairwise mode defined in this section can be used for unicast communication.

This mode uses the derivation process defined in Section 3, and allows two group members to protect requests and responses exchanged with each other using pairwise keying material.

Senders MUST NOT use the pairwise mode to protect a message addressed to multiple recipients or to the whole group. This prevents a client that wants to address one specific server from protecting a request with the pairwise key associated to that server, and then send the request over multicast.

The pairwise mode results in the same performance and security improvements displayed by optimized responses (see Section 9.2).

G.1. Pre-Requirements

In order to protect an outgoing message in pairwise mode, a sender needs to know the public key and the Recipient ID for the message recipient, as stored in its own Recipient Context associated to that recipient.

Furthermore, the sender needs to know the individual address of the message recipient. This information may not be known at any given point in time. For instance, right after having joined the group, a client may know the public key and Recipient ID for a given server, but not the addressing information required to reach it with an individual, one-to-one request.

To make this information available, servers MAY provide a resource to which a client can send a request for a server identified by its 'kid' value, or a set thereof. The specified set may be empty, hence identifying all the servers in the group. Further details of such an interface are out of scope for this document.

G.2. Pairwise Protected Request

A request in pairwise mode is protected as defined in Section 7.1, with the following differences.

- The client MUST set to 1 the sixth least significant bit of the OSCORE flag bits in the OSCORE option, i.e. the Pairwise Flag.

- The COSE_Encrypt0 object included in the request is encrypted using a symmetric pairwise key K, that the client derives as defined in Section 3. In particular, the Sender/Recipient Key is the Sender Key of the client from its own Sender Context, i.e. the Recipient Key that the server stores in its own Recipient Context corresponding to the client.

- The Counter Signature is not computed. That is, unlike defined in Section 5, the payload of the OSCORE message terminates with the encoded ciphertext of the COSE object.

Note that no changes are made to the AEAD nonce and AAD.

Upon receiving a request with the Pairwise Flag set to 1, the server MUST process it as defined in Section 7.2, with the following differences.

- No countersignature to verify is included.

- The COSE_Encrypt0 object included in the request is decrypted and verified using the same symmetric pairwise key K, that the server derives as described above for the client side and as defined in Section 3.

G.3. Pairwise Protected Response

When using the pairwise mode, the processing of a response occurs as described in Section 9.2 for an optimized response.

Appendix H. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

H.1. Version -07 to -08

- Clarified relation between pairwise mode and group communication (Section 1).

- Improved definition of "silent server" (Section 1.1).

- Clarified when a Recipient Context is needed (Section 2).

- Signature checkers as entities supported by the Group Manager (Section 2.3).

- Clarified that the Group Manager is under exclusive control of Gid and Sender ID values in a group, with Sender ID values under each Gid value (Section 2.3).

- Mitigation policies in case of recycled 'kid' values (Section 2.4).
- More generic exhaustion (not necessarily wrap-around) of sender sequence numbers (Sections 2.5 and 10.11).
- Pairwise key considerations, as to group rekeying and Sender Sequence Numbers (Section 3).
- Added reference to static-static Diffie-Hellman shared secret (Section 3).
- Note for implementation about the external_aad for signing (Sectino 4.3.2).
- Retransmission by the application for group requests over multicast as Non-Confirmable (Section 7).
- A server MUST use its own Partial IV in a response, if protecting it with a different context than the one used for the request (Section 7.3).
- Security considerations: encryption of pairwise mode as alternative to group-level security (Section 10.1).
- Security considerations: added approach to reduce the chance of global collisions of Gid values from different Group Managers (Section 10.5).
- Security considerations: added implications for block-wise transfers when using the signature mode for requests over unicast (Section 10.7).
- Security considerations: (multiple) supported signature algorithms (Section 10.13).
- Security considerations: added privacy considerations on the approach for reducing global collisions of Gid values (Section 10.15).
- Updates to the methods for synchronizing with clients' sequence number (Appendix E).
- Simplified text on discovery services supporting the pairwise mode (Appendix G.1).
- Editorial improvements.

H.2. Version -06 to -07

- Updated abstract and introduction.
- Clarifications of what pertains a group rekeying.
- Derivation of pairwise keying material.
- Content re-organization for COSE Object and OSCORE header compression.
- Defined the Pairwise Flag bit for the OSCORE option.
- Supporting CoAP Observe for group requests and responses.
- Considerations on message protection across switching to new keying material.
- New optimized mode based on pairwise keying material.
- More considerations on replay protection and Security Contexts upon key renewal.
- Security considerations on Group OSCORE for unicast requests, also as affecting the usage of the Echo option.
- Clarification on different types of groups considered (application/security/CoAP).
- New pairwise mode, using pairwise keying material for both requests and responses.

H.3. Version -05 to -06

- Group IDs mandated to be unique under the same Group Manager.
- Clarifications on parameter update upon group rekeying.
- Updated external_aad structures.
- Dynamic derivation of Recipient Contexts made optional and application specific.
- Optional 4.00 response for failed signature verification on the server.
- Removed client handling of duplicated responses to multicast requests.

- Additional considerations on public key retrieval and group rekeying.
- Added Group Manager responsibility on validating public keys.
- Updates IANA registries.
- Reference to RFC 8613.
- Editorial improvements.

H.4. Version -04 to -05

- Added references to draft-dijk-core-groupcomm-bis.
- New parameter Counter Signature Key Parameters (Section 2).
- Clarification about Recipient Contexts (Section 2).
- Two different external_aad for encrypting and signing (Section 3.1).
- Updated response verification to handle Observe notifications (Section 6.4).
- Extended Security Considerations (Section 8).
- New "Counter Signature Key Parameters" IANA Registry (Section 9.2).

H.5. Version -03 to -04

- Added the new "Counter Signature Parameters" in the 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 keying 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).

H.6. 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.

H.7. 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.

H.8. 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 Christian Amsuess, Stefan Beck, Rolf Blom, Carsten Bormann, Esko Dijk, Klaus Hartke, Rikard Hoeglund, Richard Kelsey, John Mattsson, Dave Robin, Jim Schaad, Ludwig Seitz, Peter van der Stok and Erik Thormarker 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