DICE Working Group   M. Tiloca
Internet-Draft   S. Raza
Intended Status: Standards Track   SICS Swedish ICT AB
Expires: April 16, 2016   K. Nikitin
                                                               EPFL
                                                          S. Kumar
                                                   Philips Research
                                                  October 14, 2015

           Secure Two-Way DTLS-Based Group Communication in the IoT
                   draft-tiloca-dice-secure-groupcomm-00

Abstract

   CoAP has emerged as the de-facto IoT standard for communication
   involving resource-constrained devices composing Low-power and Lossy
   Networks (LLNs). CoAP mandates the adoption of the DTLS protocol to
   secure unicast communication. However, in several IoT application
   scenarios involving a group of multiple devices, the adoption of CoAP
   multicast communication through IPv6 results in a number of
   advantages, especially in terms of performance and scalability. Yet,
   CoAP does not specify how to secure multicast group communication in
   an interoperable way. This draft presents a method to secure
   communication in a multicast group, through an adaptation of the DTLS
   record layer. In particular, group members rely on the same group
   keying material in order to secure both request messages sent via
   multicast and possible unicast messages sent as response. Since the
   group keying material is provided upon joining the group, all group
   members are not required to perform any DTLS handshake with each
   other. The proposed method makes it possible to provide either group
   authentication or source authentication of secured messages.

Status of this Memo

This Internet-Draft will expire on April 16, 2016.

Copyright and License Notice

Table of Contents

1 Introduction

Nowadays it is possible to connect the physical and cyber world by
embedding a tiny computer, with limited storage and communication
capabilities, in everyday physical objects. The resultant smart
objects can be connected not only with each other in Low-Power and
Lossy Networks (LLNs), but also with the Internet, so actively taking
part to the Internet of Things (IoT). In the IoT, smart objects can
interact with each other through different communication paradigms,
namely one-to-one, one-to-many, and many-to-many. Besides, different
communication and networking protocols are standardized to enable
interactions between smart objects. For instance, 6LoWPAN [RFC6282]
enables IP capabilities, RPL [RFC6550] enables routing capabilities,
and CoAP [RFC7252] enables web capabilities.

In particular, CoAP has become the de-facto web standard for the IoT,
and mandates the adoption of the DTLS protocol [RFC6347] to provide
secure communication, if requested. More recently, [RFC7390] has
enabled group communication for CoAP, highlighting various use cases
where smart objects benefit from a group communication model. There
are multiple use cases where secure group communication is highly
convenient or even inevitable, such as lightening control, integrated
building control, software and firmware updates, parameter and
configuration updates, commissioning of 6LoWPAN networks, and
emergency broadcasts. While in several real-world IoT deployments
such use cases require security as well, CoAP does not currently
specify how to secure multicast group communication.

Since DTLS is the mandated security protocol for unicast
communication in the IoT, it makes particular sense to extend DTLS in
order to enable secure multicast communication among smart objects.
Although IPsec multicast [RFC5374] is a possible alternative, it
would require to adopt additional heavyweight security protocols such
as IPsec [RFC4301][RFC6040] and likely even IKEv2 [RFC7296].
Moreover, since unicast communication in the IoT is supposed to be
protected through DTLS, switching to a full adoption of solutions
based on IPsec becomes even less convenient and practical.

There were previous attempts to enable secure DTLS-based group communication [I-D.keoh-dice-multicast-security-08][I-D.kumar-dice-multicast-security-00] for CoAP. However, they were questioned due to a number of reasons. First, more relevant use uses cases were expected to be provided and more strongly motivated. Second, since group authentication may be not enough for many use cases, source authentication was pointed out as a fundamental option to be available. Third, the initial lack of protection for group response messages was covered by adopting traditional DTLS unicast session. This is prone to practical issues and can result in performance degradation, especially on large scale, dynamic, groups. Following the discussions around previous proposals, there is a consensus that secure group communication is indeed necessary and desirable, but specific methods proposed so far to achieve it were questioned.

In this draft, we propose an approach providing secure two-way DTLS-based group communication in the IoT, overcoming the limitations of previous proposals. This draft particularly focuses on the following goals: i) avoiding the need to implement multiple security protocols, by extending the DTLS record layer; ii) protecting multicast request messages as well as related unicast response messages in the group; iii) making it possible to ensure source or group authentication of both group request messages and group response messages; and iv) avoiding any DTLS handshake to enable secure group communication.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This specification uses the following terminology:

o Keying material: Data that is necessary to establish and maintain a cryptographic security association. This includes, for instance, keys, key pairs, and IVs [RFC4949].

o Security Association (SA): Set of policies and cryptographic keying material that together provide security services to network traffic matching this policy [RFC3740]. In this draft, a Security Association includes, together with possible additional ones, the following attributes:

   * selectors, e.g. source and destination transport addresses;

   * properties, such as identities of involved entities;

   * cryptographic policy, e.g. the algorithms, modes, key

lifetimes, and key lengths used for the authentication or
confidentiality processes;

*   keying material used for the authentication, encryption and
    signing processes.

o   Group Security Association: A bundling of security associations
    (SAs) that together define how members of a multicast group
    communicate securely [RFC3740].

o   Group Controller (GC): Entity responsible for creating a multicast
    group, establishing security associations among authorized group
    members, and managing the joining of new group members. This
    entity may also be responsible for renewing/updating multicast
    group keys and related policies, i.e. act as group key manager.
    The GC is not required to be an actual member of the multicast
    group and to take part in the group communication.

o   Sender: Entity in a multicast group that sends data as multicast
    messages to the group. In a 1-to-N multicast group, only a single
    sender transmits data to the group; in an M-to-N multicast group
    (where M and N do not necessarily have the same value), M group
    members are senders.

o   Listener: Entity in a multicast group that receives multicast
    messages when listening to the multicast IPv6 address associated
    to the multicast group. A listener MAY reply back, by sending a
    unicast response message to the sender which has sent the
    multicast message.

o   Group request: Multicast message sent by a sender in the group to
    all listeners in the group through multicast IPv6.

o   Group response: Unicast message sent back by a listener node in
    the group as a response to a group request received by a sender.

o   Group authentication: Evidence that a received message originated
    from some member of this group. This provides assurances that the
    message was not tampered with by an adversary outside this group,
    but does not pinpoint and/or assure what specific entity in the
    group originated the message.

o   Source authentication: Evidence that a received message originated
    from a specifically identified group member. This provides
    assurances that the message was not tampered with by any other
    group member or an adversary outside this group.

1.2 **Outline**

The remainder of this draft is organized as follows. Section 2 describes some relevant group communication use cases in the IoT and identifies a set of security requirements. Section 3 overviews the proposed two-way DTLS-based secure group communication, assuming that the group members already have in their possession everything required to fully operate. Section 4 presents the security data structures referred throughout this draft. Then, Sections 5 and 6 describe the details of the adaptation of the DTLS record layer, to secure multicast request messages and unicast reply messages, respectively. Section 7 discusses considerations related to the revocation and redistribution of security material in the multicast group. Section 9 presents the security considerations.

2. Use Cases and Requirements

This section introduces some use cases for group communication in the IoT and identifies a set of related security requirements.

2.1 Group Communication Use Cases

"Group Communication for CoAP" [RFC7390] provides the necessary background for multicast-based CoAP communication, with particular reference to low-power and lossy networks (LLNs) and resource constrained devices. The interested reader is encouraged to first read this document to understand the non-security related details. This document also lists a few group communication use cases with detailed descriptions.

a. Lighting control: consider a building equipped with 6LoWPAN [RFC4944][RFC6282] IP-connected lighting devices, switches, and 6LoWPAN border routers. The devices are organized into groups according to their physical location in the building. For instance, lighting devices and switches in a room or corridor can be configured as members of a single multicast group. Switches are then used to control the lighting devices, by sending on/off/dimming commands to all lighting devices in a group. 6LoWPAN border routers that are connected to an IPv6 network backbone (which is also multicast-enabled) are used to interconnect 6LoWPAN routers in the building. Consequently, this would also enable logical multicast groups to be formed even if devices in the lighting group may be physically in different subnets (e.g. on wired and wireless networks). Group communication enables synchronous operation of a group of 6LoWPAN connected lights, ensuring that the light preset (e.g. dimming level or color) of a large group of luminaires are changed at the same perceived time. This is especially useful for providing a visual synchronicity of light effects to the user. 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.

b. 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 multicast 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 multicast group.
Furthermore, 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.

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

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

e. Commissioning of LLNs systems: a commissioning device is
responsible for querying all devices in the local network or a
selected subset of them, in order to discover their presence, and
be aware of their capabilities, default configuration, and
operating conditions. Queried devices displaying similarities in

their capabilities and features, or sharing a common physical
location can be configured as members of a single multicast 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.

f. Emergency broadcast: a particular emergency related information
   (e.g. natural disaster) is generated and broadcast by an emergency
   notifier, and relayed to multiple devices. The latters may reply
   back to the emergency notifier, in order to provide their feedback
   and local information related to the ongoing emergency.

2.2 Security Requirements

The following security requirements are out of the scope of this
draft and are assumed to be already fulfilled:

a. Establishment of a GSA: A secure mechanism must be used to
   distribute keying material, multicast security policies and
   security parameters to members of a multicast group. A GSA must be
   established by the Group Controller (which manages the multicast
   group) among the group members. The 6LoWPAN border router, a
   device in the 6LoWPAN network, or a remote server outside the
   6LoWPAN network, could play the role of the Group Controller.
   However, the GSA establishment is out of the scope of this draft,
   and it is anticipated that an activity in IETF dedicated to the
   design of a generic key management scheme for the LLN will include
   this feature preferably based on [RFC3740][RFC4046][RFC4535].

b. Multicast data security ciphersuite: All group members MUST agree
   on a ciphersuite to provide authenticity, integrity and
   confidentiality of messages in the multicast group. The
   ciphersuite is specified as part of the GSA. Typically,
   authenticity is more important than confidentiality in LLNs.
   Therefore, the approach described in this draft MUST support at
   least ciphersuites with MAC only (NULL encryption) and AEAD
   [RFC5116] ciphersuites. Other ciphersuites defined for data record
   security in DTLS SHOULD also be preferably supported.

c. Backward security: A new device joining the multicast group should
   not have access to any old GSAs used before its joining. This
   ensures that a new group member is not able to decrypt
   confidential data sent before it has joined the group. The adopted
   key management scheme should ensure that the GSA is updated to
   ensure backward confidentiality. The actual mechanism to update
   the GSA and renew the group keying material upon a group member's
   joining has to be defined as part of the group key management
   scheme.

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

The following security requirements need to be fulfilled by the
approach described in this draft:

a. Multicast communication topology: This draft considers both 1-to-N
   (one sender and multiple listeners) and M-to-N (multiple senders
   and multiple listeners) communication topologies. The 1-to-N
   communication topology is the simplest group communication
   scenario that would serve the needs of a typical LLN. For
   instance, in the lighting control use case, switches are the only
   entities responsible for sending commands to a group of lighting
   devices. In more advanced lighting control use cases, a M-to-N
   communication topology would be required, for instance in case
   multiple sensors (presence or day-light) are responsible to
   trigger events to a group of lighting devices.

b. Multicast group size: Security solutions for group communication
   SHOULD be able to adequately support different, possibly large,
   group sizes. Group size is the combination of the number of
   senders and listeners in a multicast group, with possible overlap
   (i.e. a sender MAY also be a listener at the same time). In the
   use cases mentioned in this draft, the number of senders (normally
   the controlling devices) is much smaller than the number of
   listeners (i.e. the controlled devices). A security solution for
   group communication that supports 1 to 50 senders would be able to
   properly cover the group sizes required for most use cases that
   are relevant for this draft. The total number of group members is
   expected to be in the range of 2 to 100 devices. Groups larger
   than that SHOULD be divided into smaller independent multicast
   groups, e.g. by grouping lights in a building on a per floor
   basis.

c. Data replay protection: It MUST NOT be possible to replay a group
   request message or group response message, which would disrupt the
   correct communication in the group and the activity of group
   members.

d. Group-level data confidentiality: Messages sent within the
   multicast group SHOULD be encrypted. In fact, some control

commands and/or associated responses could pose unforeseen
security and privacy risks to the system users, when sent as
plaintext. In particular, data confidentiality MAY be required if
privacy sensitive data is exchanged in the group. This draft
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 multicast group, but not by an
external adversary or other external entities.

e. Data authentication: Messages sent within the multicast group
SHOULD be authenticated. That is, it is essential to ensure that a
message is originated by a generic member of the group (group-
level data authentication) or a specific member of the group
(source-level data authentication). The approach proposed in this
draft makes it possible to provide group-level data authentication
or source-level data authentication, both for group requests
originated by sender nodes and group responses originated by
listener nodes. In case group-level data authentication is
considered, it is assumed that all group members are trusted not
to tamper with the messages sent within the group, and the common
group keying material is used to authenticate messages. In case
source-level data authentication is considered, messages are
signed by their respective originator group member by means of
public-key cryptography.

f. Data integrity: Messages sent within the multicast group SHOULD be
integrity protected. That is, it is essential to ensure that a
message has not been tampered with by an external adversary or
other external entities which are not group members. Data
integrity is provided through the same means used to provide data
authentication.

3. Overview of DTLS-based Secure Multicast

This draft describes how to adapt the DTLS protocol to secure group
communication. To this end, we propose an extension of DTLS, based on
minimal adaptation to the DTLS record layer [RFC6347]. Reusing the
DTLS protocol for different purposes can guarantee the required level
of security, while avoiding the need to implement multiple security
protocols. This is especially beneficial when the target deployment
consists of embedded, resource-constrained, devices.

DTLS has been selected as the default, must-implement, security
protocol for securing communication relying on the CoAP protocol
[RFC7252]. Therefore, it is desirable that DTLS is also accordingly
extended to secure CoAP-based group communication [RFC7390].
Nevertheless, the DTLS extension described in this draft does not
imply the usage of the CoAP protocol. That is, different protocols

can be adopted at the application level, provided that they support
the presence of DTLS.

This section first presents group communication based on IPv6
multicast, and then overviews a possible adaptation of DTLS to secure
group communication. In particular, the proposed approach makes it
possible to secure group request sent by senders as multicast
messages and related group responses sent back by listeners as
unicast messages. Also, it makes it possible to provide source
authenticity of group messages, both for group requests and group
responses. Finally, unless otherwise necessary to establish further
specific secure sessions, the proposed approach does not require to
perform any DTLS handshake between group members.

3.1 IPv6 Multicast

Group members are categorized into two possible roles, namely sender
and listener. Any group member may have one of these roles, or both
roles. The application(s) running on a group member is supposed to
determine these roles, depending on the intended interactions with
the communication stack. In principle, a sender or listener does not
require any prior access procedures or authentication to send or
listen to a multicast message [RFC5374].

```
                        ++++
                        |. |
                      --| ++++
         ++++     /   ++|. |
         |A |---------| ++++
         |  |     \     ++|B |
         ++++      \-----|  |
        Sender          ++++
                      Listeners
```
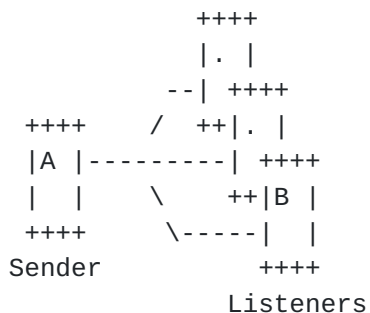
Figure 1: Example of a 1-to-N multicast communication

A sender to an IPv6 multicast group sets the destination of the
packet to an IPv6 address that has been allocated for IPv6 multicast.
A device becomes a listener by "joining" to the specific IPv6
multicast group. This in turn requires the joining device to register
with a network routing device, signaling the intent to receive
packets sent to that particular IPv6 multicast group. Figure 1
depicts an example of 1-to-N multicast communication and the roles of
the group members. In principle, any device can decide to listen to
any IPv6 multicast address. This also means that applications on the
other group members do not know, or do not get notified, when new
listeners join the multicast group. More details on the IPv6
multicast and CoAP group communication can be found in [RFC7390].

This draft does not intend to modify any of the underlying group
communication or multicast routing protocols.

3.2 Two-way Secure Group Communication

This draft assumes the presence of a group controller (GC), i.e. a
dedicated entity that creates and manages the multicast group. The
group controller may be hosted by a remote server or be a border
router. In some cases, devices that intend to join the multicast
group may be configured by means of a commissioning tool that
mediates the communication between them and the group controller.
Devices can discover the group controller by using various methods
defined in [I-D.vanderstok-core-dna] such as DNS-SD [RFC6763] and
Resource Directory [I-D.ietf-core-resource-directory]. The group
controller communicates with individual devices to add them to the
group. Additionally it provides them with the GSA, consisting of the
keying material, security policies, security parameters and
ciphersuites. To this end, the group controller relies on
standardized key management mechanisms which are out of the scope of
this draft. Additional ciphersuites may need to be defined to convey
the bulk cipher algorithm, MAC algorithm and key lengths within the
key management protocol.

A sender in the multicast group encrypts and authenticates a group
request multicast message by using the group keying material to
process the DTLS record. Then, the group request is passed down to
the lower layer of the IP protocol stack for transmission to the
multicast address as depicted in Figure 2. Upon receiving the group
request, the listeners use the multicast IP destination address and
port number (i.e., Multicast identifier) to look up the GSA
associated to that group connection. The received group request is
then decrypted and its authenticity is verified. More details about
how group requests are processed in the multicast group are provided
in Section 5.

A listener MAY reply back to the sender that originated the group
response, by means of a unicast group response message. The listener
secures the group response by means of individual keying material
derived from the group keying material in the GSA and used to process
the DTLS record. Then, the group response is passed down to the lower
layer of the IP protocol stack for transmission to the unicast
address of the sender, as depicted in Figure 2. Upon receiving the
group response, the sender considers the group keying material in the
GSA, and uses it to derive the individual keying material associated
to the listener that has sent the group response. The received group
response is then decrypted and its authenticity is verified. More
details about how group responses are processed in the multicast
group are provided in Section 6.

```
+--------+---------------------------------------------------+
|        | +--------+----------------------------------+ |
|        | |        | +------------+----------------+ | |
|        | |        | |            | +------------+ | | |
|   IP   | |  UDP   | | DTLS Record | |    Group   | | | |
| header | | header | |   Header   | |   message  | | | |
|        | |        | |            | +------------+ | | |
|        | |        | | +------------+----------------+ | |
|        | | +--------+----------------------------------+ |
+--------+---------------------------------------------------+
```
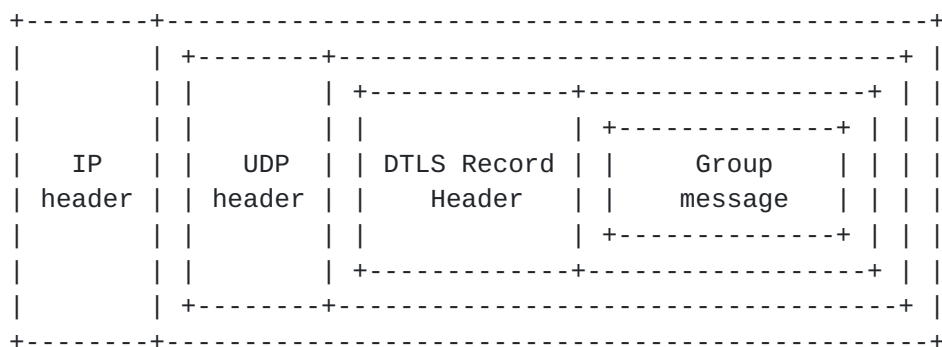
          Figure 2: A message protected using the group DTLS Record Layer

   While assuring two-way secure communication within the multicast
   group, the approach described in this draft also displays the
   following benefits. First, the commonly shared group keying material
   makes it possible to avoid the performance of DTLS handshakes
   altogether, unless otherwise necessary to establish pairwise secure
   sessions among group members. As a consequence, it does not
   substantially affect performance of group members (especially if
   senders), with further benefits in terms of network readiness and
   availability. Second, senders can remain unaware of current and
   future listener nodes in the multicast group. Third, listener nodes
   are able to securely reply back to sender nodes, without performing a
   DTLS handshake first. This is aligned with the guidelines for CoAP
   group communication [RFC7252], according to which the endpoint acting
   as the CoAP client (i.e. the sender node) SHOULD also act as the DTLS
   client (i.e. start the DTLS handshake).

4. Security Data Structures

   This section overviews the security data structures referred by the
   approach proposed in this draft. In particular, Section 4.1 discusses
   the Group Security Association (GSA) owned by all members of the
   multicast group. Then, Section 4.2 overviews the security parameter
   structure. Finally, Section 4.3 discusses the connection states
   referred by senders or listeners in the multicast group.

4.1 Group Security Association

   The GSA contains the following elements. Some of them can be not set
   or not relevant, as explained below. Every node owns the GSA of each
   multicast group where it is a member.

```
     GroupID
     SenderID
     CipherSuite
     client write IV
```

```
        server write IV
        client write encryption key
        server write encryption key
        client write MAC key
        server write MAC key
        source_authentication
        Member Private Key
        {SenderID1, SenderID1 Public Key}
        {SenderID2, SenderID2 Public Key}
        ...
        {Listener1 IP address, Listener1 Public Key}
        {Listener2 IP address, Listener2 Public Key}
        ...
```

a. The GroupID has the same value for all the group members, and
    practically represents a short alias for the multicast IP address
    associated to the multicast group. In the approach described in
    this draft, the GroupID assumes a value in the range between 0 and
    255 included.

b. The SenderID value is relevant only for group members configured
    (also) as senders, and is provided by the GC upon joining the
    group. A SenderID value MUST be unique within the multicast group
    at any time, i.e. the multicast group MUST NOT include more than
    one member configured (also) as sender and associated to a given
    SenderID value. In the approach described in this draft, a
    SenderID assumes a value in the range between 0 and 255 included.
    The GC provides a list of active sender nodes and their respective
    SenderID to all the listener nodes in the multicast group.

c. The GC chooses a CipherSuite which is specified in the GSA of all
    the members of the multicast group. If the flag
    source_authenticity is set to TRUE, CipherSuite MUST support
    source authentication based on public key cryptography. All the
    members of the multicast group MUST support the CipherSuite
    specified in the GSA.

d. The same group keying material client write IV, server write IV,
    client write encryption key, server write encryption key, client
    write MAC key and server write MAC key is derived for all members
    of the multicast group. To this end, all group members consider
    the security parameters master_secret[48], client_random[32] and
    server_random[32] in Section 4.2, and rely on the PRF function
    defined in Section 6.3 of [RFC5246] to derive the group keying
    material. Every listener node, before processing its first group
    response message addressed to a given sender node, derives
    additional individual keying material and stores it separately
    (see Section 6). From then on, the listener node considers this

individual keying material as the client write encryption key and
client write MAC key used when securing outgoing group response
messages addressed to that sender node (see Section 6). That is,
the client write encryption key and client write MAC key in the
GSA on the listener nodes are not affected. When receiving a group
request message from that listener node for the first time, that
sender node derives the same individual keying material in order
to process incoming group response messages from that sender node
(see Section 6), and stores it separately. That is, the client
write encryption key and client write MAC key in the GSA on the
sender nodes are not affected.

e. The source_authentication flag is set to TRUE if it is required to
   provide source authentication of all multicast group request
   messages sent by all sender nodes and all unicast group response
   messages sent by all listener nodes. Conversely, the
   source_authentication is set to FALSE if it is required to provide
   only group authentication to all multicast group request messages
   sent by all sender nodes and all unicast group response messages
   sent by all listener nodes.

f. If the source_authentication flag is set to TRUE, every member of
   the multicast group stores its own asymmetric private key in the
   Member Private Key field. Conversely, if the source_authentication
   flag is set to FALSE, the Member Private Key field is neither set
   nor relevant on any member of the multicast group.

g. If the source_authentication flag is set to TRUE, every listener
   node stores the public key of each sender node in the multicast
   group, together with the respective SenderID. In principle,
   listener nodes can be provided with the senders' public keys upon
   joining the group. As an alternative, a listener node can ask a
   trusted Certification Authority for a sender's public key, upon
   receiving a multicast group request message from that sender node
   for the first time. More details about the retrieval and
   verification of senders' public keys are out of the scope of this
   draft.

h. If the source_authentication flag is set to TRUE, every sender
   node stores the public key of each listener node in the multicast
   group, together with the respective IP address. In principle,
   sender nodes can ask a trusted Certification Authority for a
   listener's public key, upon receiving a unicast group response
   message from that listener node for the first time. More details
   about the retrieval and verification of senders' public keys are
   out of the scope of this draft.

4.2 Security parameter structure

The Group Controller provides the following security parameter
structure to all members of the multicast group, upon their joining.
This structure reflects the definition in [RFC5246], with three
additional fields in order to provide source authentication in the
group, if required.

```
struct {
        ConnectionEnd           entity;
        PRFAlgorithm            prf_algorithm;
        BulkCipherAlgorithm     bulk_cipher_algorithm;
        CipherType              cipher_type;
        uint8                   enc_key_length;
        uint8                   block_length;
        uint8                   fixed_iv_length;
        uint8                   record_iv_length;
        MACAlgorithm            mac_algorithm;
        uint8                   mac_length;
        uint8                   mac_key_length;
        SignatureAlgorithm      signature_algorithm;
        uint8                   signature_key_length;
        uint8                   signature_length;
        CompressionMethod       compression_algorithm;
        opaque                  master_secret[48];
        opaque                  client_random[32];
        opaque                  server_random[32];
    } SecurityParameters;
```

a. SecurityParameters.entity is set to ConnectionEnd.server for
   sender nodes and ConnectionEnd.client for listener nodes.

b. The parameters bulk_cipher_algorithm, cipher_type, enc_key_length,
   block_length, fixed_iv_length, record_iv_length, mac_algorithm,
   mac_length, mac_key_length, signature_algorithm,
   signature_key_length and signature_length are set to the same
   value for all the members in the multicast group, based on the
   ciphersuite specified in the GSA. In particular, the parameters
   signature_algorithm, signature_key_length and signature_length are
   set only if the flag source_authentication in the GSA is set to
   TRUE. Also, the parameters mac_algorithm, mac_length and
   mac_key_length are set only if the flag source_authentication in
   the GSA is set to FALSE and the ciphersuite specified in the GSA
   supports message authenticity.

c. The parameters prf_algorithm, compression_algorithm,
   master_secret[48], client_random[32], and server_random[32] are
   set to the same value for all the members in the multicast group.

4.3 Connection states

The group connection states are instantiated on the group members
based on their role(s) in the group, according to the following
rules.

A group member configured as sender instantiates the following
group connection states:

o One write group connection state, instantiated during the sender
   node's initialization and referred while sending multicast group
   request messages. This group connection state contains an Epoch
   value and a Sequence Number value which is incremented for each
   DTLS record sent as part of an outgoing multicast group request
   message. The first of such DTLS records has 0 as Sequence Number
   value.

o One read group connection state for each listener node in the
   multicast group. This kind of group connection state is
   instantiated upon receiving a unicast group response message from
   the associated listener node for the first time. Then, it is
   referred when receiving any further unicast group response message
   from the associated listener node. This group connection state
   contains an Epoch value and a Sequence Number value, which is
   updated according to the DTLS anti-replay mechanism, upon
   receiving a valid unicast group response message from the
   associated listener node. The Sequence Number value is initialized
   to 0. Also, the group connection state contains additional
   individual keying material related to the associated listener
   node, generated and used as described in Section 6.

A group member configured as listener instantiates the following
group connection states:

o One write group connection state for each sender node in the
   multicast group, referred while sending unicast group response
   messages to the associated sender. This kind of group connection
   state is instantiated upon sending a unicast group response
   message to the associated sender node for the first time. Then, it
   is referred when sending any further unicast group response
   message to the associated sender node. This group connection state
   contains an Epoch value and a Sequence Number value which is
   incremented for each DTLS record sent as part of an outgoing
   unicast group response message to the associated sender node. The
   first of such DTLS records has 0 as Sequence Number value. Also,
   this write group connection state contains additional individual
   keying material related to the associated sender node, generated
   and used as described in Section 6.

o One read group connection state for each sender node in the

multicast group. This kind of group connection state is
instantiated upon receiving a multicast group request message from
the associated sender node for the first time. Then, it is
referred when receiving any further multicast group request
message from the associated sender node. This group connection
state contains an Epoch value and a Sequence Number value, which
is updated according to the DTLS anti-replay mechanism, upon
receiving a valid multicast group request message from the
associated sender node. The Sequence Number value is initialized
to 0.

5. Secure multicast messages

This section describes the adaptation of the DTLS Record layer to
enable multiple sender nodes in the group to securely send multicast
group request messages.

```
+---------+---------+--------+--------+------------+--------+
| 1 Byte  | 2 Byte  | 2 Byte | 1 Byte |   5 Byte   | 2 Byte |
+---------+---------+--------+--------+------------+--------+
| Content | Version | Epoch  | Sender | Trunc_seq_ | Length |
|  Type   | Ma | Mi |        |   ID   |   number   |        |
+---------+---------+--------+--------+------------+--------+
```
 Figure 3: Adapted DTLS record header for multicast group requests

Figure 3 shows the adapted DTLS record layer header, used when a
sender node transmits a multicast group request message. The existing
DTLS record layer header is adapted in such a way that the 6-octet
Sequence_number field is split into a 1-octet SenderID field and a 5-
octet "truncated" Trunc_seq_number field. The epoch is fixed and
provided to the group members by the GC upon joining the multicast
group. The Trunc_seq_number is initialized to 0 and is increased by
one each time the sender node sends a new DTLS record, as part of a
multicast group request message. The group request message is secured
by the source sender node as described in Section 5.2, and processed
by the recipient listener nodes as described in Section 5.3.

5.1 On truncating the sequence number field

The rationale for having the Trunc_seq_number field together with the
SenderID field is as follows.

In the presence of multiple sender nodes in the multicast group, the
sequence number values used by sender nodes need to be synchronized
to avoid their reuse. Otherwise, multicast group request messages
sent by different sender nodes may get discarded as replayed messages
by the recipient listener nodes. Moreover, since all the sender nodes
refer to the same group keying material, this would also result in

nonce reuse in AEAD cipher suites like AES-CCM [RFC6655] and AES-GCM [RFC5288]. In particular, nonce reuse can completely break the security of these cipher suites. In fact, according to the AES-CCM specification for TLS [RFC6655], the CCMNonce is a combination of a salt value and the message sequence number, that is

```
                    struct {
                        opaque salt[4];
                        opaque nonce_explicit[8];
                    } CCMNonce;
```

More in detail, the salt is the "client write IV" (when the client is sending) or the "server write IV" (when the server is sending). Besides, [RFC6655] states that the value of the nonce_explicit MUST be distinct for each distinct invocation of the CCM encrypt function for any fixed key. When the nonce_explicit is equal to the sequence number of the TLS packets, the CCMNonce has the following structure.

```
        struct {
                uint32 client_write_IV; // low order 32-bits
                uint64 seq_num;          // TLS sequence number
        } CCMClientNonce;

        struct {
                uint32 server_write_IV; // low order 32-bits
                uint64 seq_num;          // TLS sequence number
        } CCMServerNonce;
```

In DTLS, the 64-bit sequence number is composed of the 16-bit epoch value concatenated with the 48-bit sequence number value included in the DTLS record header. Therefore, in order to prevent that the CCMNonce is reused, either all the sender nodes in the multicast group are synchronized with each other, or separate non-overlapping sequence number spaces have to be created for each sender node. Synchronization between sender nodes is particularly difficult to achieve, especially among constrained devices and in LLNs. Therefore, this draft considers the second approach and separates the sequence number spaces by embedding a unique sender identifier, namely SenderID, in the sequence number, as suggested in [RFC5288]. It follows that the GC is also required to assign a unique SenderID to each device in the multicast group which is configured as sender. The GC provides a list of active sender nodes and their SenderID to all listener nodes in the multicast group.

5.2 Sending secure group request messages

In order to send a DTLS record as part of a secure multicast group request message, a sender node proceeds as follows.

1. The sender node determines the right GSA and write group
   connection state, based on the multicast IP destination address
   and destination port number.

2. The sender node includes its own SenderID in the SenderID field of
   the adapted header of the DTLS record.

3. The sender node considers the epoch and truncated sequence number
   values from the write group connection state. These values are
   used to fill the Epoch and Trunc_seq_number fields of the adapted
   header of the DTLS record. The first record sent to the multicast
   group has truncated sequence number 0.

4. The sender node processes the DTLS record according to the
   CipherSuite specified in the GSA. In particular, if the
   source_authentication flag in the GSA is set to TRUE, the DTLS
   record is signed using the sender's private key, and the signature
   is appended to the DTLS record payload. Otherwise, if the
   CipherSuite provides only group authentication, the server write
   MAC key and server write IV stored in the GSA are used to
   authenticate the DTLS record. Finally, if the CipherSuite provides
   confidentiality, the server write encryption key and server write
   IV stored in the GSA are used to encrypt the DTLS record.

5. The sender node passes the DTLS record down to UDP and IP layers
   for transmission on the multicast IP destination address and port
   number associated to the multicast group.

6. The sender node updates the truncated sequence number in its own
   write group connection state, incrementing its value by 1.

5.3 Receiving secure group request messages

   Upon receiving a DTLS record as part of a secure multicast group
   request message, a listener node proceeds as follows.

1. The listener node determines the right GSA, based on the multicast
   IP destination address and destination port number.

2. The listener node considers the multicast IP destination address
   and destination port number as well as the SenderID specified in
   the adapted header of the DTLS record, in order to determine the
   right read group connection state. This is different from the
   standard DTLS logic, according to which the current "client read"
   connection state is bound to the IP source address and port
   number. If this is the first group request message ever received
   from that sender node in the multicast group, the listener node
   instantiates a read group connection state associated to that

sender node.

3. The listener node checks the freshness of the received DTLS
   record, comparing the values conveyed in the Epoch and
   Trunc_seq_number fields with the values of the epoch and truncated
   sequence number stored in the selected read group connection
   state. The listener node MUST ensure that the retrieved epoch
   value coincides with the stored value, and that the retrieved
   truncated sequence number value coincides with the stored value.
   Otherwise, the DTLS record SHOULD be discarded. Alternatively, a
   mechanism based on a sliding window MAY be adopted to accept
   genuine out-of-order DTLS records.

4. The listener node processes the DTLS record according to the
   CipherSuite specified in the GSA. If the CipherSuite provides
   confidentiality, the server write encryption key and server write
   IV stored in the GSA are used to decrypt the DTLS record. Then, if
   the source_authentication flag in the GSA is set to TRUE, the
   listener node checks the authenticity of the DTLS record using the
   sender's public key, identified by means of the SenderID conveyed
   in the record header. Otherwise, if the CipherSuite provides only
   group authentication, the server write MAC key and server write IV
   stored in the GSA are used to check the authenticity of the DTLS
   record.

5. Once the freshness and authenticity of the DTLS record have been
   verified, the listener node passes the DTLS record to the higher
   level protocols.

6. The listener node updates the value of the truncated sequence
   number in the selected read group connection state.

[6](). Secure unicast responses to multicast messages

This section describes the adaptation of the DTLS Record layer to
enable listener nodes in the multicast group to securely send unicast
group response messages, as a reply to a multicast group request
message received from a sender node. Unless otherwise necessary to
fulfill further application requirements, the listener nodes are not
required to perform any DTLS handshake with other members of the
multicast group.

Figure 4 shows the adapted DTLS record layer header, used when a
listener node transmits a unicast group response message. The
existing DTLS record layer header is adapted in such a way that the
6-octet Sequence_number field is split into a 1-octet GroupID field
and a 5-octet "truncated" Trunc_seq_number field. The epoch is fixed
and provided to the group members by the GC upon joining the

multicast group. The Trunc_seq_number is initialized to 0 and is
increased by one each time the listener node sends a new DTLS record,
as part of a unicast group response message addressed to the same
sender node.

```
+---------+---------+--------+--------+------------+--------+
| 1 Byte  | 2 Byte  | 2 Byte | 1 Byte |   5 Byte   | 2 Byte |
+---------+---------+--------+--------+------------+--------+
| Content | Version | Epoch  | Group  | Trunc_seq_ | Length |
|  Type   | Ma | Mi |        |   ID   |   number   |        |
+---------+---------+--------+--------+------------+--------+
```
Figure 4: Adapted DTLS record header for unicast group responses

The rationale for having the Trunc_seq_number field together with the
GroupID field is as follows. The GroupID field allows a sender node
to correctly "interpret" a unicast group response message as a reply
to a multicast group request message that it has previously sent to
that multicast group. Practically, the GroupID field allows a sender
node to correctly determine the correct GSA to be considered. Also,
together with the IP source address of a unicast group response
message, the GroupID field allows a sender node to correctly
determine the correct read group connection state to be considered,
i.e. the one referred to the correct listener node. The case where a
sender node is configured as sender in different multicast groups
having the same GroupID is discussed in Section 9.6.

Potentially, the same issues discussed in Section 5.1 about the reuse
of nonce values can arise. In fact, all listener nodes in the
multicast group refer to the same GroupID, and it is possible that
their truncated sequence number values become synchronized with each
other. In order to preserve secure communication within the multicast
group, listener nodes derive additional individual keying material
which is used to process outgoing unicast group response messages.
The same individual keying material is derived by sender nodes and
used to process incoming unicast group response messages. This is
described in detail in Section 6.1, while further related security
considerations are provided in Sections 9.2 and 9.3. Group response
messages are secured by a source listener node as described in
Section 6.2, and processed by the recipient sender node as described
in Section 6.3.

6.1 Derivation of listener individual keying material

Before sending its first unicast group response message to a given
sender node in the multicast group, a listener node computes
additional individual keying material, and stores it in the write
group connection state associated to that sender node and the same
group's communication, as detailed below. From then on, the listener

node considers that individual keying material to process all its
outgoing unicast group response messages addressed to that specific
recipient sender node.

Similarly, upon receiving the first group response message from that
listener node in the multicast group, the sender node computes the
same individual keying material, and stores it in the read group
connection state associated to that listener node and the same
group's communication. From then on, the sender node considers that
individual keying material to process all the incoming unicast group
response messages received from that listener node and related to
that group's communication.

The individual keying material is computed according to the same data
expansion scheme referred by DTLS and described in Section 6.3 of
[RFC5246]. The invoked procedure considers as input the client write
key and server write key originally stored in the GSA, the IP address
associated to the listener node, and the SenderID associated to the
sender node. In particular, the following procedure is performed

```
    key_block = PRF(client write key + server write key,
                    "key derivation",
                    listener IP address + SenderID);
```

until enough output has been generated. Specifically, the procedure
stops when an amount of bytes equal to
SecurityParameters.mac_key_length plus
SecurityParameters.enc_key_length has been generated. Then, the
individual keying material is extracted as follows.

The first SecurityParameters.mac_key_length bytes of the key_block
output are considered as the client write MAC key to be used for
unicast responses sent by that listener node to that sender node.
That is, the listener node stores it as the individual client write
MAC key in the write group connection state associated to that sender
node and the same group's communication. Instead, the sender node
stores it as the individual client write MAC key in the read group
connection state associated to that listener node and the same
group's communication.

Then, the following SecurityParameters.enc_key_length of the
key_block output are considered as the client write encryption key to
be used for unicast responses sent by that listener node to that
sender node. That is, the listener node stores it as the individual
client write encryption key in the write group connection state
associated to that sender node and the same group's communication.
Instead, the sender node stores it as the individual client write
encryption key in the read group connection state associated to that

listener node and the same group's communication.

6.2 Sending secure group response messages

In order to send a DTLS record as part of a secure unicast group
response message, a listener node proceeds as follows. If it has
previously established a traditional unicast DTLS session with the
sender node to be replied to, the listener node MUST refer to that
DTLS session to securely reply to the sender node. In such a case,
the listener node is not required to maintain a write group
connection state associated to that sender node and that group's
communication. Otherwise, the listener node performs the following
steps.

1. The listener node determines the right GSA, based on the multicast
   IP destination address and destination port number of the
   multicast group request message it has received and wants to reply
   to.

2. If this is its first group response message ever sent to that
   sender node in the multicast group, the listener node instantiates
   a write group connection state associated to that sender node.
   Then, the listener node derives the necessary individual keying
   material and stores it in the write group connection state, as
   described in Section 6.1. Otherwise, the listener node determines
   the right write group connection state based on the unicast IP
   destination address and destination port number of the sender node
   to be replied to. The IP destination address is the one previously
   retrieved as IP source address from the multicast group request
   message. The destination port number is the one previously
   retrieved as source port number from the multicast group request
   message.

3. The listener node retrieves the GroupID from the GSA and includes
   it in the GroupID field of the adapted header of the DTLS record.

4. The listener node considers the epoch and truncated sequence
   number values from the selected write group connection state.
   These values are used to fill the Epoch and Trunc_seq_number
   fields of the adapted header of the DTLS record. The first record
   sent to the recipient sender node has truncated sequence number 0.

5. The listener node processes the DTLS record according to the
   CipherSuite specified in the GSA. In particular, if the
   source_authentication flag in the GSA is set to TRUE, the DTLS
   record is signed using the listener's private key, and the
   signature is appended to the DTLS record payload. Otherwise, if
   the CipherSuite provides only group authentication, the individual

client write MAC key stored in the selected write group connection
state and the client write IV stored in the GSA are used in order
to authenticate the DTLS record. Finally, if the CipherSuite
provides confidentiality, the individual client write encryption
key stored in the selected write group connection state and the
client write IV stored in the GSA are used to encrypt the DTLS
record.

6. The listener node passes the DTLS record down to UDP and IP layers
   for transmission on the unicast IP destination address and port
   number of the sender to be replied to. The IP destination address
   is the one previously retrieved as IP source address from the
   multicast group request message. The destination port number is
   the one previously retrieved as source port number from the
   multicast group request message.

7. The listener node updates the truncated sequence number in the
   selected write group connection state, incrementing its value by
   1.

6.3 Receiving secure group response messages

Upon receiving a DTLS record as part of a secure unicast group
response message, a sender node proceeds as follows. A node in the
multicast group MUST consider invalid and discard any received group
response message, in case it is not configured as sender node.

1. The sender node checks if a unicast DTLS session has been
   previously opened with the source listener node, based on the IP
   source address and port number of the received group response
   message. In case of positive match, the sender node MUST process
   the received message according to the traditional DTLS protocol
   [RFC6347]. Note that, in such a case, the sender node is not
   required to maintain a read group connection state associated to
   that listener node and that group's communication. Conversely, in
   case of negative match, the sender node does not yet consider the
   received message to be invalid, and moves to step 2.

2. The sender node parses the DTLS record header according to the
   format shown in Figure 4, and determines the right GSA, based on
   the retrieved GroupID. In case the sender node is configured as
   sender in different multicast subgroups having the same GroupID,
   the right GSA can be determined as discussed in Section 9.6. If no
   GSA associated to that GroupID can be found, the sender node MUST
   discard the received message. Otherwise, the sender node moves to
   step 3.

3. If this is the first unicast group response message ever received

from that listener node in the multicast group, the sender node
instantiates a read group connection state associated to that
listener node. Then, the sender node derives the necessary
individual keying material and stores it in that read group
connection state, as described in Section 6.1. Otherwise, the
sender node considers the retrieved GroupID and the IP source
address of the listener node, in order to determine the right read
group connection state.

4. The sender node checks the freshness of the received DTLS record,
   comparing the values conveyed in the Epoch and Trunc_seq_number
   fields with the values of the epoch and truncated sequence number
   stored in the selected read group connection state. The sender
   node MUST ensure that the retrieved epoch value coincides with the
   stored value, and that the retrieved truncated sequence number
   value coincides with the stored value. Otherwise, the DTLS record
   SHOULD be discarded. Alternatively, a mechanism based on a sliding
   window MAY be adopted to accept genuine out-of-order DTLS records.

5. The sender node processes the DTLS record according to the
   CipherSuite specified in the GSA. If the CipherSuite provides
   confidentiality, the sender node decrypts the DTLS record using
   the individual client write encryption key stored in the selected
   read group connection state and the client write IV stored in the
   GSA. Then, if the source_authentication flag in the GSA is set to
   TRUE, the sender node checks the authenticity of the DTLS record
   using the listener's public key, identified by means of the
   listener IP address. Otherwise, if the CipherSuite provides only
   group authentication, the sender node checks the authenticity of
   the DTLS record using the individual client write MAC key stored
   in the selected read group connection state and the client write
   IV stored in the GSA.

6. Once the freshness and authenticity of the DTLS record have been
   verified, the sender node passes the DTLS record to the higher
   level protocols.

7. The sender node updates the value of the truncated sequence number
   in the selected read group connection state.

7. Revocation and redistribution of group security material

The approach for secure group communication presented in this draft
SHOULD rely on additional mechanisms aimed at securely distributing,
revoking, and renewing keying material, multicast security policies,
and security parameters used in the group. In addition, the GC is
intended as primarily responsible for providing the GSA to the group
members. The establishment of the GSA will be part of a future IETF

activity dedicated to the design of a generic key management scheme,
preferably based on requirements and recommendations defined in
[RFC3740][RFC4046][RFC4535]. Nevertheless, in this section we suggest
a possible way to renew the group security material (rekeying) and
provide the current GSA to new group members upon their joining. In
particular, we consider the GC to be an additional member of the
multicast group, and to be configured as a sender node.

As specified in [RFC5246], the group keying material included in the
GSA is computed from the Security Parameters master_secret,
client_random and server_random. in particular, the master_secret is
computed by means of a pre_master_secret value, and the Security
Parameters client_random and server_random. Hence, a group keying
material renewal can be practically performed by securely providing
the group members with a new pre_master_secret value. Upon receiving
it, the group members use it to compute the new master_secret to be
stored in the SecurityParameters structure. Finally, they use the new
master_secret to compute the new group keying material to be stored
in the GSA, as well as the individual keying material to be stored in
the write group connection states on the listener nodes and in the
read group connection states on the sender nodes. All the other
Security Parameters and keying material in the GSA remain unchanged.

In order to assure that rekeying messages are actually sent by the
GC, it is required that source authentication is enabled in the
multicast group, i.e. the source_authentication flag in the GSA is
set to TRUE. As an alternative, the GC can determine the random field
of the new pre_master_secret as the next element of a pre-computed
reversed hash chain, an authentication mechanism derived from
Lamport's one-time password [LAMPORT:1981]. The advantage of this
approach is that the most recently released element in the chain can
be efficiently authenticated by computing its hash, and verifying
that the result is equal to the previously released element in the
chain. Therefore, upon creating the multicast group, it is sufficient
that the GC provides all members with the head element of the hash
chain in an authenticated way, e.g. off-line or through a predefined
point-to-point authenticated channel. Then, all other chain elements
can be automatically and efficiently authenticated by group members.
Furthermore, this approach does not require to configure the GC as an
actual member of the group.

What follows is a brief overview of the key management operations
expected on a regular basis (Section 7.1), and upon a new member's
joining (Section 7.2) or a group member's leaving (Section 7.3).

7.1 Periodical rekeying.

The group keying material SHOULD be renewed in a periodical fashion,

in order to discourage an external adversary from breaking the secure communication in the group through exhaustive key search or traffic analysis. Periodical rekeying is effective as long as the group has not been compromised, i.e. no group members are under the adversary's control. Furthermore, the amount of time between two consecutive occurrences of periodical rekeying SHOULD be appropriately defined, taking into account the specific application requirements, and the expected level of threat the group is exposed to. Obviously, the more frequently the periodical rekeying is performed, the less the damage due to possibly compromised group keying material.

Practically, the GC can periodically broadcast a new securely generated pre_master_secret to group members. For instance, this can be done by broadcasting a single DTLS multicast message, protected by means of the current server write parameters in the GSA, as any other multicast group request message transmitted to the group. Upon receiving it, all group members rely on the new pre_master_secret to update the master_secret in the SecurityParameters structure. After that, all group members can update the group security material stored in their GSA, as well as the individual keying material stored in the write group connection states on the listener nodes and in the read group connection states on the sender nodes.

7.2 Join rekeying.

Upon a new node joining the group, it is required to assure backward security, whose importance has been explicitly stressed in [RFC4046]. In particular, the joining node must not be able to access any group communication which took place before its joining. As a first step, currently present group members are rekeyed according to the same procedure described above for periodical rekeying. After that, the GC provides the updated GSA to the joining node, which can then complete the join process. The GC is required to provide the joining node with the GSA through a secure communication channel. As a possible approach, upon contacting the Resource Directory service to gain knowledge of the GC address, the joining node could retrieve a public certificate associated to the GC, use the retrieved public key to establish a secure channel with the GC, and securely obtain the GSA. As an alternative, the joining node can establish a DTLS session with the GC, and receive the GSA as a sequence of protected DTLS records.

7.3 Leave rekeying.

A group member may voluntarily leave the group (i.e. its mission is concluded or its membership is expired), or it may be forced to leave (e.g. if it has been found to be compromised or it is suspected so). In such a case, remaining group members have to be securely provided with updated group keying material, in order to assure forward

security, whose importance has been explicitly stressed in [RFC4046].
Specifically, the leaving node must not be able to access group
communication which takes place after its departure from the group.
At the same time, the leaving node must be prevented from taking part
in the rekeying process itself, i.e. from accessing the new security
material during its distribution. Since the leaving node is aware of
the current group keying material, the latter can not be used to
securely distribute a new pre_master_secret to the remaining nodes.
As a first easy approach, the GC can establish a different pairwise
symmetric key with every member of the group upon their joining.
Then, upon a node's leaving, the GC could rely on such pairwise keys
to distribute a new pre_master_secret to all the remaining nodes, in
a one-to-one fashion. However, such a unicast approach lacks of
efficiency, as it requires a number of rekeying messages which
linearly grows with the number of nodes in the group, hence not
scaling well with the group size. As an alternative, it is possible
to rely on specific application level schemes for group key
management, which result to be more efficient and display high
scalability with the number of nodes in the group.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

This document describes an adaptation of DTLS 1.2 to support
multicast group communication; therefore, most of the security
considerations are the same as those of DTLS 1.2 [RFC6347] and TLS
1.2 [RFC5246] described in Appendices D, E, and F. Besides, security
issues specifically related to the revocation and redistribution of
security material are discussed in Section 7. Further security issues
to be taken into consideration are discussed below.

9.1 Group-level security

The approach described in this draft relies on commonly shared group
security material to protect communication within the group. Unless
source authentication is adopted, this requires that all group
members are trusted, e.g. they do not forge messages in order to make
them appear as sent by a different member of the group. In many use
case, the devices in a multicast group belong to a common authority
and are configured by a commissioner. For instance, in a professional
lighting scenario, the roles of the sender and listener nodes are
configured by the lighting commissioner, and devices follow those
roles. In case group members can not be trusted to this extent, it is
possible to provide source authenticity through digital signature, as
described in Sections 5 and 6.

Furthermore, encryption and authentication operations rely on a nonce
value in order to randomize the process. With reference to a listener
node which secures a unicast response message, it composes the nonce
as the sequence of i) the client write IV from the GSA; and ii) the
concatenation of epoch, GroupID, and Trunc_seq_number, namely
explicit nonce, that are going to be included in the response DTLS
header. On one hand, the client write IV and the GroupID are the same
for all the listeners in the group. On the other hand, it is possible
that multiple listeners are synchronized with each other as to the
epoch and Trunc_seq_number they use to reply to a given multicast
message from a sender node. In this case, all such listeners consider
the same explicit nonce. Sections 9.2 and 9.3 discuss that reuse of
explicit nonce does not represent a problem, since individual
security material associated to listener nodes is derived on the
listener and sender side.

Finally, the presented approach SHOULD take into consideration the
risk of compromise of group members. As a first thing, the risk of
compromise is reduced when multicast groups are deployed in
physically secured locations, like lighting inside office buildings.
Secondly, the adoption of key management schemes for secure
revocation and renewal of group security material SHOULD be
considered, as discussed in Section 7.

9.2 Prevention of attacks based on IP spoofing

The approach described in this draft makes it possible that all the
listener nodes in the multicast group use different individual pairs
<client write encryption key, client write MAC key> (see Section 6).
It follows that, when securing a unicast group response message, two
listener nodes that rely on the same explicit nonce actually use
different keying material and hence produce two different secured
group response messages. This makes it possible to prevent an active
attack when an external adversary injects fake group response
messages by spoofing the IP source address of a given listener node.

Furthermore, every listener node derives an individual pair <client
write encryption key, client write MAC key> separately for each
sender node in the group (see Section 6.1). This makes it possible to
prevent an active attack when an external adversary intercepts a
unicast group response message sent to a given sender node, and then
replays it to a different sender node by spoofing the IP destination
address.

9.3 Nonce reuse in authenticated encryption

DTLS makes it possible to use Authenticated Encryption with

Associated Data (AEAD) ciphers, such as AES_CCM [RFC6655] or AES_GCM [RFC5288]. These ciphers require only one key to be used to secure a message, i.e. listeners would rely only on the client write encryption key, without using the client write MAC key.

In this case, the possible reuse of the same explicit nonce by different listener nodes discussed in Section 9.1 is not a problem, thanks to the derivation of different individual client write encryption keys for each listener node (see Section 6). This has two major consequences. First, it makes it possible to still comply with the requirement of [RFC6655] and [RFC5288], according to which each value of the explicit nonce MUST be distinct for each distinct invocation of the encrypt function for any fixed key. Second, it prevents an adversary from solving underlying keyed hash so making subsequent forgeries trivial, and from choosing IVs to produce colliding counters. It follows that any degradation of the provided security level due to a possible nonce reuse is prevented.

9.4 Late joining nodes

Upon joining the multicast group when the system is fully operative, listener nodes are not aware of the current epoch and Trunc_seq_number being used by different sender nodes. This means that, when such listener nodes receive a multicast message from a sender node, they are not able to verify if the message is fresh and has not been replayed by an adversary. In order to address this issue, we can rely on techniques similar to AERO [I-D.mcgrew-aero], i.e., upon receiving the first message from a particular sender node, late joining listener nodes initialize their last seen epoch and Trunc_seq_number in their read group connection state associated to that sender node. However, after that they drop such a message, without delivering it to the application layer. This provides a reference point to identify if future messages are fresher than the last one seen. As an alternative, the GC can be an additional member of the multicast group and be configured as a listener node. Then, it can maintain the epoch and Trunc_seq_number of each sender node in the multicast group. When late joiners send a request to the GC to join the multicast group, the GC can provide them with the list of epoch and Trunc_seq_number values to be stored in the read group connection states associated to the appropriate senders.

9.5 Uniqueness of SenderIDs

As discussed in Section 5, it is important that SenderIDs are unique within a multicast group, in order to preserve the security properties of the DTLS record layer messages, especially the freshness of nonce values. Hence, in case two or more sender nodes are configured with the same SenderID, it becomes necessary to

explicitly avoid the related security weakness and prevent nonce
reuse. To this end, a possible mechanism consists in all sender nodes
in the multicast group acting also as listener nodes. This allows a
sender node S1 that receives a message with the same SenderID in the
DTLS header from a different sender node S2 to become aware of the
SenderID conflict. Then, S1 can notify the GC about the SenderID
conflict through a secure channel. The GC can then provide a
different SenderID to either one of the two sender nodes or both of
them.

9.6 Same GroupID in different multicast groups

In general, every multicast group is managed by a different GC, which
is responsible to determine the associated GroupID to be used within
the group. Then, it is possible that a node is configured as sender
in different multicast groups at the same time, and such multicast
groups are identified by the same, ambiguous, GroupID value. In such
a case, upon receiving a unicast group response message as a reply to
a multicast group request message, such sender nodes would not be
able to determine to which group communication such group response
message refers to, i.e. which GSA and read group connection state has
to be considered to process the received group response message. In
order to overcome this issue, sender nodes can identify the right GSA
and read group connection state to be considered by means of the
extended tuple <GroupID, IP source address, destination port number>
from the DTLS, UDP and IP headers in the group response message. This
relies on the practical assumption that, if a sender node belongs to
different multicast groups at the same time, it actually takes part
in each group's communication through a different application or
application instance, each one of which refers to a different port
number.

9.7 Reduced sequence number space

In the approach described in this draft, the DTLS record layer
sequence_number is truncated from 6 octets to 5 octets. This results
in a reduction of the sequence_number space, which SHOULD be taken
into account to ensure that the epoch value is incremented before the
Trunc_seq_number wraps around. A sender node, or the GC as an
alternative, can send a ChangeCipherSpec message to the whole
multicast group whenever the Trunc_seq_number has been exhausted, in
order that the epoch value is increased by all group members. This
SHOULD be done as part of the key management scheme adopted within
the group, which is out of scope for this draft.

10. References

10.1 Normative References

[I-D.keoh-dice-multicast-security-08]
          Keoh, S., Kumar, S., Dijk, E., and A. Rahman, "DTLS-based
          Multicast Security in Constrained Environments", draft-
          keoh-dice-multicast-security-08 (work in progress), July
          2014.

[I-D.kumar-dice-multicast-security-00]
          Kumar, S., and R. Struik, "Transport-layer Multicast
          Security for Low-Power and Lossy Networks (LLNs)", draft-
          kumar-dice-multicast-security-00 (work in progress), March
          2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
          Encryption", RFC 5116, January 2008.

[RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security
          (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois
          Counter Mode (GCM) Cipher Suites for TLS", RFC 5288,
          August 2008.

[RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer
          Security Version 1.2", RFC 6347, January 2012.

[RFC6655] Bailey, D., and D. McGrew, "AES-CCM Cipher Suites for
          Transport Layer Security (TLS)", RFC 6655, July 2012.

[RFC7252] Shelby Z., Hartke K., and C. Bormann, "The Constrained
          Application Protocol (CoAP)", RFC 7252, June 2014.

10.2 Informative References

[LAMPORT:1981]
          Lamport, L., Password authentication with insecure
          communication. Communications of the ACM, 24(11):770-772,
          November 1981.

[I-D.mcgrew-aero]
          McGrew, D., and J. Foley, "Authenticated Encryption with
          Replay prOtection (AERO)", draft-mcgrew-aero-01 (work in
          progress), February 2014.

[I-D.ietf-core-resource-directory]
          Shelby, Z., Koster, M., Bormann, C., and P. van der Stok,

                    "CoRE Resource Directory", draft-ietf-core-resource-
                    directory-04 (work in progress), July 2015.

     [I-D.vanderstok-core-dna]
                    Stok, P., Lynn, K., and A. Brandt, "CoRE Discovery,
                    Naming, and Addressing", draft-vanderstok-core-dna-02
                    (work in progress), July 2012.

     [RFC3740]   Hardjono, T., and B. Weis, "The Multicast GroupSecurity
                    Architecture", RFC 3740, March 2004.

     [RFC4046]   Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm,
                    "Multicast Security (MSEC) Group Key Management
                    Architecture", RFC 4046, April 2005.

     [RFC4301]   Kent, S., and K. Seo, "Security Architecture for the
                    Internet Protocol", RFC 4301, December 2005.

     [RFC4535]   Harney, H., Meth, U., Colegrove, A., and G. Gross,
                    "GSAKMP: Group Secure Association Key Management
                    Protocol", RFC 4535, June 2006.

     [RFC4944]   Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
                    "Transmission of IPv6 Packets over IEEE 802.15.4
                    Networks", RFC 4944, September 2007.

     [RFC4949]   Shirey, R., "Internet Security Glossary, Version 2", RFC
                    4949, August 2007.

     [RFC5374]   Weis, B., Gross, G., and D. Ignjatic, "Multicast
                    Extensions to the Security Architecture for the Internet
                    Protocol", RFC 5374, November 2008.

     [RFC6040]   Briscoe, B., "Tunnelling of Explicit Congestion
                    Notification", RFC 6040, November 2010.

     [RFC6282]   Hui, J., and P. Thubert, "Compression Format for IPv6
                    Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,
                    September 2011.

     [RFC6550]   Winter T., Thubert P., Brandt A., Hui J., Kelsey R., Levis
                    P., Pister K., Struik R., Vasseur JP., and R. Alexander,
                    "RPL: IPv6 Routing Protocol for Low-Power and Lossy
                    Networks", RFC 6550, March 2012.

     [RFC6763]   Cheshire, S., and M. Krochmal, "DNS-Based Service
                    Discovery", RFC 6763, February 2013.

   [RFC7296]   Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T.
               Kivinen, "Internet Key Exchange Protocol Version 2
               (IKEv2)", RFC 7296, October 2014.

   [RFC7390]   Rahman., A., and E. Dijk, "Group Communication for the
               Constrained Application Protocol (CoAP)", RFC 7390,
               October 2014.

Authors' Addresses

   Marco Tiloca
   SICS Swedish ICT AB
   Isafjordsgatan 22, 16440 Kista
   Sweden

   Email: marco@sics.se

   Shahid Raza
   SICS Swedish ICT AB
   Isafjordsgatan 22, 16440 Kista
   Sweden

   Email: shahid@sics.se

   Kirill Nikitin
   School of Computer and Communication Sciences, EPFL
   EDOC-IC INN 134, Station 14, 1015 Lausanne
   Switzerland

   Email: kirill.nikitin@epfl.ch

   Sandeep S. Kumar
   Philips Research
   High Tech Campus 34, Eindhoven 5656 AE
   The Netherlands

   Email: sandeep.kumar@philips.com