A Group Keying Protocol
<draft-ietf-trill-group-keying-00.txt>

Abstract

   This document specifies a general group keying protocol. It also
   provides use profiles for the application of this group keying
   protocol to multi-destination TRILL Extended RBridge Channel message
   security and TRILL over IP packet security.

Status of This Memo

Table of Contents

1. Introduction

This document specifies a general group keying protocol in Section 2. In addition, it provides, in Section 3, the use profile for the application of this group keying protocol to a case using DTLS (TRILL [RFC6325] [RFC7780] Extended RBridge Channel message security [RFC7178] [RFC7978]) and IPsec [TRILLoverIP}. It is anticipated that there will be other uses for this group keying protocol.

1.1 Terminology and Acronyms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology and acronyms defined in [RFC6325] and [RFC7178]. Some of these are repeated below for convenience along with additional new terms and acronyms.

AES - Advanced Encryption Standard.

Data Label - VLAN or FGL.

DTLS - Datagram Transport Level Security [RFC6347].

FGL - Fine Grained Label [RFC7172].

GKd - A distinguished station in a group that is in charge of which group keying (Section 2) is in use.

GKs - Stations in a group other than GKd (Section 2).

HKDF - Hash based Key Derivation Function [RFC5869].

IS-IS - Intermediate System to Intermediate System [RFC7176].

keying material - The set of a Key ID, a secret key, and a cypher suite.

PDU - Protocol Data Unit.

QoS - Quality of Service.

RBridge - An alternative term for a TRILL switch.

SHA - Secure Hash Algorithm [RFC6234].

TRILL - Transparent Interconnection of Lots of Links or Tunneled
   Routing in the Link Layer.

TRILL switch - A device that implements the TRILL protocol
   [RFC6325] [RFC7780], sometimes referred to as an RBridge.

2. Group Keying Protocol

   This section defines a general Group Keying Protocol that provides
   shared secret group keys. Any particular use of this protocol will
   require profiling giving further details and specifics for that use.
   The protocol is not suitable for discovery messages but is intended
   for use between members of a group that have already established
   pair-wise security.

2.1 Assumptions

   The following are assumed:
      - All pairs of stations in the group can engage in pairwise
        communication with unicast messages and each can groupcast a
        message to the other group members.
      - At any particular time, there is a distinguished station GKd in
        the group that is in charge of keying for the groupcast data
        messages to be sent to the group. The group wide shared secret
        keys established by GKd are referred to herein as "dynamic"
        keys.
      - Pairwise keying has been negotiated between GKd and each other
        station GKs1, GKs2, ... GKsN in the group. These keys are
        referred to in this protocol as "pairwise" keys.
      - One or more keys, other than the dynamic or pairwise keys, each
        of which is already in place at all group member stations. These
        are referred to as "stable" keys.

   When keying material is stored by a station, it is accompanied by a
   "use flag" indicating whether or not that keying material is usable
   for groupcast transmissions.

2.2 Group Keying Procedure Overview

   GKd sends unicast keying messages to the other stations in the group
   and they respond as specified below and in further detail in the
   particular use profiles for this Group Keying Protocol. All such
   keying messages MUST be encrypted and authenticated using the
   pairwise keys as further specified in the use profile.

   Typically, GKd sends a keying message to each GKs with keying
   material. After successful acknowledgement of receipt from each GKs,
   GKd sends a keying message to each GKs instructing it to use the
   dynamic key GKd has set. It would be common for GKd to set a new
   dynamic key at each GKs while an older dynamic key is in use so that
   GKd can more promptly roll over to the new key when appropriate.

To avoid an indefinite build up of keying material at a GKs, keys
have a lifetime specified by GKd and GKd can send a message deleting
a key. (GKd can also send a message indicating that a key is no
longer to be used but leaving it set.) Should the space available at
a GKs for keying material be exhausted, on receipt of a Set Key
keying message for a new key ID GKs discards a dynamic key it has and
originates a Delete Key message to the source of that dynamic key.

2.3 Transmission and Receipt of Group Data Messages

If a group has only two members, then pairwise security is used
between them.

When a group has more than two members and a station in the group
transmits a data message to the group, if the transmitter has one or
more keys set by GKd that it has been instructed to use, it uses one
of those keys and its associated cypher suite to groupcast the data
message. If it has no such key, then it uses serial unicast to send
the data message to each other member of the group, negotiating
pairwise keys with them if it does not already have such pairwise
keys. Thus it is a responsibility of GKd not to authorize the use of
a groupcast key until it knows that all the GKs have that key.

When a station in the group receives data that has been groupcast to
the group, if the receiver has the key referenced by the data message
the receiver decrypts and verifies it. If verification fails or if
the receiver does not have the required key, the receiver discards
the data message. Thus whether GKs has been directed to "use" a key
by GKd is relevant only to transmission, not reception.

2.4 Changes in Group Membership or GKd

When a new station joins the group, GKd should send that station the
currently in-use group key and instruct it to use that key and send
it other keys known to the group members and intended for future use.

If GKd detects that one or more stations that were members of the
group are no longer members of the group, it SHOULD generate and
distribute a new group key to the remaining group members, instruct
them to use this new key, and delete from them any old keys known to
the departed group member station(s) or at least instructing them to
disuse such old keys that are marked for use; however, in the case of
groups with large and/or highly dynamic membership, where a station
might frequently leave and then rejoin, it may, as a practical
matter, be necessary to rekey less frequently.

A new group member can become GKd due to the previous GKd leaving the
group or a configuration change or the like. A GKs MUST NOT use
keying material set by a station that it determines is not GKd. To
avoid a gap in service, a station that is not GKd MAY set keying
material at other stations in the group; however, such a non-GKd
station cannot set the use flag for any such keying material. It is
RECOMENDED that the second highest priority station to be GKd set
such keying material at all other stations in the group. Should a
station run out of room for keying material, it SHOULD discard keying
material set by a station with lower priority to be GKd before
discarding keying material set by a higher priority station and among
keys set by GKd is SHOULD discard the lest recently used first.

2.5 Group Keying Messages

   Keying messages start with a Version number. This document specifies
   Version zero.

   Keying messages are structured as
      o a Version number,
      o a Response flag,
      o a Key ID length,
      o the Key ID of a stable key,
      o a group keying use profile identifier,
      o possible padding, and finally
      o an AES key wrapped [RFC5649] [RFC3394] vector of additional
         fields wrapped using the stable key identified and using
         AES-256, as shown in Figure 2.1 below.

   Keying messages are always sent unicast and encrypted and
   authenticated with the appropriate pairwise key, all as further
   specified for the particular use profile. It will typically be
   possible for GKd to calculate the keying message once, including the
   AES wrapping under a stable key, then send that message to various
   GKs using the different pairwise keys for each GKs.

```
+-+-+-+-+-+-+-+-+
|Ver|R|KeyID1Lng|                 1 byte
+-+-+-+-+-+-+-+-+...
| KeyID1  ...                     KeyID1Lngth bytes
+-+-+-+-+-+-+-+-+...
| Use Type      |                 1 byte
+-+-+-+-+-+-+-+-+
| Pad1 Length   |                 1 byte
+-+-+-+-+-+-+-+-+...
| Padding                         Pad1 Length bytes
+-+-+-+-+-+-+-+-+...
|AES Wrap Length|                 1 byte
+-+-+-+-+-+-+-+-+...
|
| AES Wrapped Material            Variable size
|
+-+-+-+-+-+-+-+-...
```
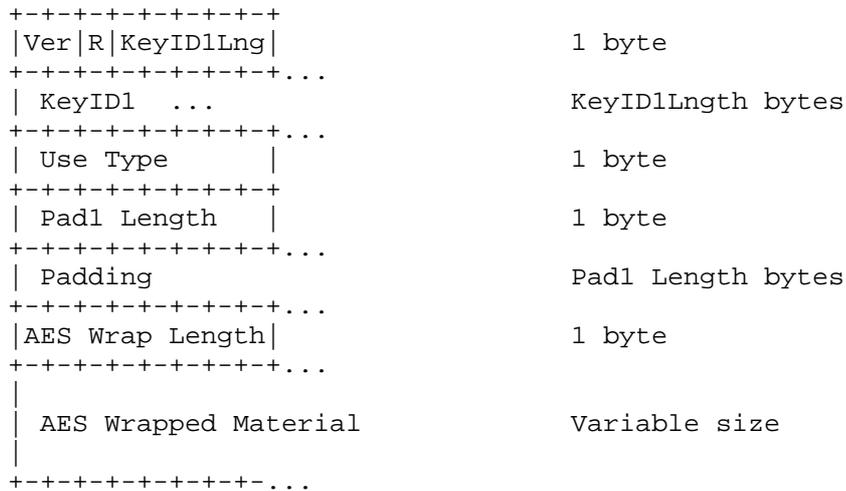
                   Figure 2.1. Keying Message Structure

     The fields in Figure 2.1 are as follows:

          Ver - Group Keying protocol version. This document specifies
               version zero.

          R - Response flag. If set to one, indicates a response message.
               If set to zero, indicated a request or no-op message.

          KeyID1Lngth, KeyID1 - KeyID1 identifies the stable AES-256 key
               wrapping key (also known as the Key Encrypting Key (KEK))
               as further specified in the use profile. KeyID1Lngth is a
               5-bit field that gives the length of KeyID1 in bytes as an
               unsigned integer.

          Use Type - Specifies the particular group security use profile
               such as RBridge Extension (Section 3) or IP link
               [TRILLoverIP].

          Pad1 Length, Pad1 - Padding to obscure the non-padded message
               size. Pad1 Length may be from 0 to 255 and gives the length
               of the padding as an unsigned integer. Each byte of padding
               MUST be equal to Pad1 Length. For example, 3 bytes of
               padding with length is 0x03030303.

          AES Wrap Length - An unsigned byte that gives the length of the
               AES Wrapped Material in units of 8 bytes. The length of AES
               key wrapped material is, as specified in [RFC5649], always
               a multiple of 8 bytes (64 bits) and not less than 16 bytes.
               Thus an AES Wrap Length of 0 or 1 is invalid.

AES Wrapped Material - The output of the AES Key Wrapping
operation on the message vector of fields using the
specified stable key.

The vector of fields contained within the AES-256 key wrapping is
specified for the various keying messages in subsections below. The
contents of this wrapped vector are protected by the AES wrapping as
well as being authenticated and super-encrypted by the pairwise keyed
security used for sending the overall keying message. The stable key
used for AES wrapping MUST be different from the outer message
pairwise key.

Each group keying message contains, in the AES wrapped vector of
fields, a message type and a message ID set by the sender of a
request. These fields are returned in the corresponding response to
assist in the matching of response to requests, except that there is
no response to the No-Op message.

If no response is received to a request (other than a No-Op message)
for an amount of time configurable in milliseconds from 1 to (2**15
- 1), the request is re-transmitted with the same message ID. These
retries can occur up to a configurable number of times from 1 to 8.
Unless otherwise provided in the particular use profile, the default
response delay threshold is 200 milliseconds and the default maximum
number of retries is 3.

Keying messages are sent with a priority/QoS configurable on a per
device per use type basis. The default priority/QoS is specified in
the use profile.

Since the minimum length of the AES Wrapped Material is 16 bytes
[RFC5649], the minimum valid size of a keying message is 20 bytes,
even if KeyID1 Length and Pad1 Length are zero. All multi-byte fields
are in network order, that is, with the most significant byte first.

2.6 Set Key Message

The structure of the wrapped vector of fields for the Set Key keying
message is as show in Figure 2.2. A recipient automatically
determines the overall length provided for this vector of fields
inside the AES wrapping as a byproduct of the process of AES
unwrapping [RFC5649].

```
+-+-+-+-+-+-+-+-+
| Msg Type = 1  |                    1 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Msg ID                             3 bytes     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Pad2 Length   |                    1 bytes
+-+-+-+-+-+-+-+-+...
| Padding                            Pad2 Length bytes
+-+-+-+-+-+-+-+-+...
| Other                              Variable size
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Lifetime                       |   2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| KeyID2 Length |                    1 byte
+-+-+-+-+-+-+-+
| KeyID2  ...                        KeyID2 Length bytes
+-+-+-+-+-+-+-+
| CypherSuiteLng|                    1 byte
+-+-+-+-+-+-+-+-+
| CypherSuite ...                    CypherSuiteLng bytes
+-+-+-+-+-+-+-+-+-...
| Key  ...                           Variable size
+-+-+-+-+-+-+-+-...
```

                Figure 2.2. Set Key Message Inner Structure

    The fields are as follows:

      Msg Type = 1 for Set Key message

      Msg ID - A 3 byte quantity to be included in the corresponding
          response message to assist in matching requests and
          responses. Msg ID zero has a special meaning in responses
          and MUST NOT be used in a Set Key message or any other
          group keying request message.

      Pad2 Length, Pad2 - Padding to obscure the size of the unapdded
          AES wrapped data. Pad2 Length may be from 0 to 255 and
          gives the length of the padding as an unsigned integer.
          Each byte of padding MUST be equal to Pad1 Length. For
          example, 2 bytes of padding with length byte is 0x020202.

      Other - Additional information if specified in the use profile.
          If Other information in this message is not mentioned in
          the use profile, there is none and this portion of the
          wrapped information is null. If a use profile specifies
          Other information it must be possible to determine its
          length so that following fields can be properly parsed and
          so that the size of the Key field can be deduced; for
          example, it could begin with a length byte.

Lifetime - A 2-byte unsigned integer. After that number of
    seconds plus one second, the key and associated information
    being set MUST be discarded. Unless otherwise specified for
    a particular use profile of this group keying protocol, the
    default Lifetime is 15,000 seconds or a little over four
    hours.

KeyID2 Length, KeyID2 - KeyID2 identifies the group key and
    associated information being set as further specified in
    the use profile. KeyID2 Length is an unsigned byte that
    gives the length of KeyID2 in bytes.

CypherSuiteLng, CypherSuite - CypherSuite identifies the cypher
    suite associated with the key being set as further
    specified in the use profile. CypherSuite Length is an
    unsigned byte the gives the length of CypherSuite in bytes.

Key - This is the actually group shared secret keying material
    being set. Its length is deduced from the overall length of
    the vector of fields (found by the AES unwrap operation)
    and the length of the preceding fields.

If GKs already has a dynamic key set under KeyID2, the key's value
and associated cypher suite are compared with those in the Set Key
messages. If they are the same, the only receiver action is to update
the Lifetime information associated with KeyID2 and send a Response
message. If they are different, the lifetime, cypher suite, and key
(and possibly Other material) are replaced, the use flag is cleared,
and a Response message sent.

2.7 Use, Delete, Disuse, or Deleted Key Messages

The structure of the wrapped material for the Use Key, Delete Key,
and Disuse Key keying messages are the same as each other except for
the message type. This structure is shown in Figure 2.3

```
+-+-+-+-+-+-+-+-+
| Msg Type = t  |                       1 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Msg ID                               3 bytes      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Pad2 Length   |                       1 bytes
+-+-+-+-+-+-+-+-+...
| Padding                              Pad2 Length bytes
+-+-+-+-+-+-+-+-+...
| Other                                Variable size
+-+-+-+-+-+-+-+-+...
| KeyID2 Length |                       1 byte
+-+-+-+-+-+-+-+-+
| KeyID2  ...                          KeyID2 bytes
+-+-+-+-+-+-+-+-+
```

        Figure 2.3. Use, Delete, Disuse, or Deleted Key Message

    The Msg Type field specifies the particular message as follows:

    Msg Type   Message
    --------   ----------
        2   Use Key
        3   Delete Key
        4   Disuse Key
        5   Deleted Key

The remaining fields are as specified in Section 2.4. KeyID2
indicates the key to be used, deleted, for which use should cease, or
which has been deleted, depending on the message type.

It is RECOMMENDED that these messages be padded so as to be the same
length as a typical Set Key message.

The Delete Key is sent by a station believing itself to be GKd
instructing some GKs to delete a key. When a GKs spontaneously
deletes a key, it sends a Deleted Key message to the station from
which it received the key. The message types for Delete Key and
Deleted Key are different to minimize confusion in corner cases such
as the GKd changing while messages are in flight. The Msg ID used in
a Deleted Key message is created by the sending GKs from a space of
Msg IDs associated with that GKs which is independent of the Msg IDs
used in requests originated by GKd.

2.8 Response Message

    The structure of the wrapped material for the Response group keying
    message is as show below in Figure 2.4. A response message is

indicated by the R bit in the first byte of the message outside the
key wrapping.

A response MUST NOT be sent due to the receipt of a response. The R
bit is outside of the key wrapping so that this rule can be enforced
even in cases of difficulty in unwrapping.

```
   +-+-+-+-+-+-+-+-+
   | Msg Type = n  |                    1 byte
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Msg ID                         3 bytes      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Pad2 Length   |                    1 bytes
   +-+-+-+-+-+-+-+-+...
   | Padding                         Pad2 Length bytes
   +-+-+-+-+-+-+-+-+...
   | Other                           Variable size
   +-+-+-+-+-+-+-+-+...
   | Response Code |                    1 byte
   +-+-+-+-+-+-+-+-+
   | ReqPartLength |                    1 byte
   +-+-+-+-+-+-+-+-+...
   | Request Part                    ReqPartLenth bytes
   +-+-+-+-+-+-+-+-...
```
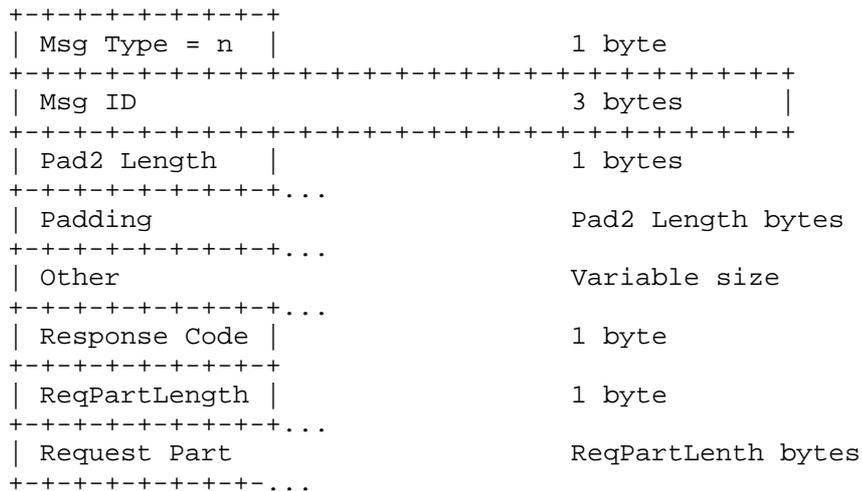
              Figure 2.4. Response Message Inner Structure

Except as specified below, the fields are as specified for the Key
Set message.

     Msg Type, Msg ID - The content of these field is copied from
          the message in reply to which this Response message is sent
          unless there is an error that stops the replying station
          from determining them; in that case the special value zero
          is used for the Msg Type and Msg ID. Errors where the Msg
          Type and ID could not be determined are indicated by a
          Response Code with its high order bit set to one, that is,
          the 0b1xxxxxxx bit set.

     Response Code - An unsigned byte giving the response as
          enumerated in Table 2.2 in Section 2.8.1. Any Response
          Code other than a success indicates that the receiver took
          no action on the request other than sending an error
          Response message.

     ReqPartLength, Request Part: It is usually usefully to include
          some or all of the request message in error responses.
          - If the Response Code high order two bits are zero, the
             request succeeded and ReqPartLength MUST be set to zero
             so Request Part will be null.

    - If the Response Code high order two bits are zero one
      (0b01xxxxxx), then there was an error in the part of the
      request inside the AES key wrapping but the unwrap
      process was successful. ReqPartLength is the length of
      the request message material included in the Request
      Part field. The included request material is from the
      unwrapped vector of fields started with the Msg Type
      byte.
    - If the Response Code high order bit is one (the
      0b1xxxxxxx is set), then there was an error parsing the
      material outside the AES key wrap or an error in the AES
      unwrapping process. ReqPartLength is the length of the
      request message part included in the Request Part field.
      The included part of the request starts with the first
      byte of the message (the byte containing the version,
      response flag, and KeyID1 Length).

2.8.1 Response Codes

   The high order two bits of the Response Code have meaning as shown in
   Table 2.1.

Top 2 Bits	Category
0b00	Success
0b01	AES wrap contents
0b10/11	Outside of AES wrap contents

Response Decimal	Response Hex	Meaning
0	0x00	Success
1	0x01	Success and the key at an existing key ID was changed
2-47	0x02-0x2F	Unassigned
48-63	0x30-0x3F	Reserved for special success codes defined in use profiles
64	0x40	Malformed inner fields (see Note 2 below)
65	0x41	Unknown or zero Msg Type in a request
66	0x42	Zero Msg ID in a request
68	0x43	Invalid length KeyID2
69	0x44	Unknown KeyID2
70	0x45	Invalid length CypherSuite
71	0x46	Unknown CyperSuite
72	0x47	Bad Key (see Note 3 below)
73-111	0x49-0x6F	Unassigned
112-127	0x70-0x7F	Reserved for error codes defined in use profiles and related to the AES wrapped

```
                           contents
      128      0x80    Malformed message (see Note 1 below)
      129      0x81    Invalid length KeyID1
      130      0x82    Unknown KeyID1
      131      0x83    Unknown Use Type
      131      0x84    AES unwrap fails test 1, see Section 3
                           [RFC5649]
      132      0x85    AES unwrap fails test 2, see Section 3
                           [RFC5649]
      133      0x86    AES unwrap fails test 3, see Section 3
                           [RFC5649]
  134-175  0x86-0x7F   Unassigned
  176-191  0xB0-0xBF   Reserved for error codes defined in use
                           profiles and related to parts of
                           message outside the AES wrap contents
      192      0xC0    No keys set
      193      0xC1    Referenced key unknown
      194      0xC2    Referenced key known but use flag not set
  195-255  0xC3-0xFF   Reserved
```

Response Code Notes:

   Note 1 Message is too short or too long, AES wrapped material is too
           short, Padding bytes are not the required value, or similar
           fundamental message format problems.

   Note 2 The AES wrapped inner vector of fields is too short or too
           long, Padding bytes are not the required value, or similar
           fundamental vector of fields format problems.

   Note 3 Key is not a valid length for CypherSuite or other internal
           checks on key (for example, parity bits in a 64 bit DES key
           (not that you should be using DES)) fail.

2.8 No-Op Message

   The No-Op message is a dummy message intended for use in disguising
   metadata deducable from keying message transmissions. It requires no
   response although a recipient can always decided to send a No-Op
   message to a station from which it has received such a message. The
   vector of fields inside the AES key wrap is as follows:

```
+-+-+-+-+-+-+-+-+
| Msg Type = 6  |                    1 byte
+-+-+-+-+-+-+-+-+
| Pad2 Length   |                    1 bytes
+-+-+-+-+-+-+-+-+...
| Padding                            Pad2 Length bytes
+-+-+-+-+-+-+-+-...
```

Figure 2.5. No-Op Message Inner Structure

The Msg Type is set to 6 to indicate a No-Op message.

Pad2 Length and Padding are as specified in Section 2.6. It is
RECOMMENDED that Pad2 Length in a No-Op message be such as to make
its length the same as the length of a typical Set Key message.

2.9 General Security Considerations

   This section gives some general security considerations of this group
   keying protocol as distinguished from security considerations of a
   particular use profile.

   The method by which the stations in the group discover each other is
   specified in the group keying use profile. GKd controls group access
   and generally learns whatever it needs to know about GKs during the
   pairwise authentication and pairwise keying process.

   The group keying provided by this protocol is shared secret keying.
   This means that data messages can only be authenticated as coming
   from some group member but not as coming from a specific group
   member. If this level of authentication is insufficient, GKd can
   simply not set keys or not set them as usable. This will force all
   stations in the group that are configured to use security for multi-
   destination transmissions to the group to serial unicast data to the
   other group members using pairwise keying.

   The content value of padding fields in the Group Keying protocol is
   fixed so that it cannot be used as a covert channel. The length of
   padding could still be so used.

3. DTLS: Extended RBridge Channel Group Keyed Security

   This section specifies a profile of the group keying protocol defined
   in Section 2. This profile provides shared secret keying to secure
   multi-destination Extended RBridge Channel messages [RFC7978]. The
   keys put in place by the group keying protocol are available for use
   as DTLS pre-shared keys with the DTLS and Composite Security of
   multi-destination Extended RBridge Channel messages as specified in
   Section 3.2.

   For this group keying use profile, a group is identified by TRILL
   Data Label (VLAN or FGL [RFC7172]) and consists of the data reachable
   [RFC7780] RBridges with interest in that Data Label. GKd is the
   RBridge in the group that, of those group members supporting the
   Group Keying Protocol, is the highest priority to be a TRILL
   distribution tree root. If not all members of the group support the
   Group Keying Protocol, then there are two cases for multi-destination
   Channel Tunnel RBridge Channel messages:
   (1) If the sender and at least two other group members support the
       Group Keying Protocol, it SHOULD, for efficiency, send a secured
       multi-destination RBridge Channel message to cover the group and
       serially unicast to the group members not supporting the Group
       Keying Protocol.
   (2) In other cases the sender serially transmits the data to the
       group members using pairwise security.

3.1 Transmission of Group Keying Messages

   Keying messages themselves are sent as unicast Extended RBridge
   Channel messages carrying a Group Keying protocol (see Section 5.2)
   RBridge Channel message. They MUST use DTLS Pairwise or Composite
   (STypes 2 or 3) security.

   The Group Keying profile for this Group Keying Use Type is as
   follows:

      Priority of Group Keying messages for this SHOULD be 6 unless the
          network manager chooses to use a lower priority after
          determining that such lower priority group keying messages
          will yield acceptable performance. Priority 7 SHOULD NOT be
          used as it may cause interference with the establishment and
          maintenance of adjacency.

      Use Type = 1

      KeyID1 Length = 2, KeyID1 is an [RFC5310] key ID.

      CypherSuiteLng = 2, CypherSuite is the cypher suite used in

groupcast extended RBridge Channel data messages for the
corresponding KeyID2. This a DTLS [RFC6347] cypher suite.

KeyID2 Length = 1, KeyID2 is the index under which a group key is
set. Group keys are, in effect, indexed by this KeyID2 and
the nickname of the GKd as used in the Ingress Nickname
field of the TRILL Header of Group Keying messages.

3.2 Transmission of Protected Multi-destination Data

Protected Extended RBridge Channel [RFC7978] messages are multicast
(M bit set to one in the TRILL Header) and set the SType field to a
new value for "Group Secured" (See Section 5.3). The data is
formatted as one byte of Key ID followed by data formatted as TLS 1.2
[RFC5246] application_data using the cyphersuite and keying material
stored under the Key ID.

4. TRILL Over IP Group Keyed Security

   This section specifies a profile of the group keying protocol defined
   in Section 2. This profile provides shared secret keying to secure
   TRILL over IP messages [TRILLoverIP]. The keys put in place by the
   group keying protocol are available for use as IPSEC keys.

   For this group keying use profile, a group is identified by an IP
   multicast address and consists of the adjacent [RFC7177] RBridges
   reachable with that multicast address. GKd is the RBridge in the
   group that, of those group members supporting the Group Keying
   Protocol, has the highest priority to be a TRILL distribution tree
   root. If not all members of the group support the Group Keying
   Protocol, then there are two cases for multi-destination TRILL over
   IP messages:
   (1) If the sender and at least two other group members support the
       Group Keying Protocol, it SHOULD, for efficiency, send a secured
       IPSEC message to cover the group and serially unicast to the
       group members not supporting the Group Keying Protocol.
   (2) In other cases the sender serially transmits the data to the
       group members using pairwise security.

4.1 Transmission of Group Keying Messages
   tbd

      Use Type = 2

   tbd

4.2 Transmission of Protected Multi-destination Data

   tbd

5. IANA Considerations

   This section gives IANA Considerations.

5.1 Group Keying Protocol

   IANA is requested to perform the following actions:

      1. Establish a protocol parameters web page for "Group Keying
         Protocol Parameters" with the initial registries on that page
         as specified below in this section.

      2. Establish a "Message Type" registry on the Group Keying
         Protocol Parameters page as follows:

         Registration Procedure: IETF Review

         Reference: [this document]

            Type   Description   Reference
            -------   -----------   ---------------
                  0   Reserved   [This document]
                  1   Set Key   [This document]
                  2   Use Key   [This document]
                  3   Delete Key   [This document]
                  4   Disuse Key   [This document]
                  5   Deleted Key   [This document]
                  6   No-Op   [This document]
            7-250   Unassigned
          251-254   Reserved for Private Use [This document]
                255   Reserved   [This document]

      3. Establish a "Group Keying Use Profile" registry on the Group
         Keying Protocol Parameters page as follows:

         Registration Procedure: IETF Review

         Reference: [This document]

            Profile   Description   Reference
            --------- -----------   ---------
                  0   Reserved   [This document]
                  1   Extended RBridge Channel [This document]
                  2   TRILL over IP   [This document]
            3-250   Unassigned
          251-254   Reserved for Private Use [This document]
                255   Reserved   [This document]

     4. Establish a "Response Code" registry on the Group Keying
        Protocol Parameters page as show below taking entries from the
        Response Code table in Section 2.8.1 above. In the table of
        values, the Reference column should be "[This document]" except
        where the Meaning is "Unassigned" or "Reserved".

        Registration Procedure: IETF Review

        Reference: [This document]

        Note: The top two bits of the Response Code indicate a category
        as specified in Section 2.8.1 of [this document].

          Response   Response
           Decimal   Hex   Meaning   Reference
          --------   --------- -----------   ---------
                 0   0x00   Success   [this document]
               ...   .
               255   0xFF   Reserved

5.2 Group Keying RBridge Channel Protocol Numbers

   IANA is requested to assign TBD1 as the TRILL RBridge Channel
   protocol number, from the range assigned by Standards Action, for use
   when the "Group Keying" protocol is transmitted over Extended RBridge
   Channel messages.

   The added RBridge Channel protocols registry entry on the TRILL
   Parameters web page is as follows:

          Protocol Description   Reference
          -------- -------------   ------------------
            TBD1   Group Keying   Section 2 of [this document]

5.3 Group Secured Extended RBridge Channel SType

   IANA is requested to assign TBD2 as the Group Secured SType in the
   "Extended RBridge Channel Security Types Subregistry" on the TRILL
   Parameters web page as follows:

          SType   Description   Reference
          -----   ------------- ----------
           TBD2   Group Secured Section 3.2 of [this document]

6. Security Considerations

   TBD

   See [RFC7978] for Extended RBridge Channel security.

   See [RFC7457] in connection with TLS and DTLS security.

Normative References

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

    [RFC3394] - Schaad, J. and R. Housley, "Advanced Encryption Standard
         (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394,
         September 2002, <http://www.rfc-editor.org/info/rfc3394>.

    [RFC5246] - Dierks, T. and E. Rescorla, "The Transport Layer Security
         (TLS) Protocol Version 1.2", RFC 5246, August 2008,
         <http://www.rfc-editor.org/info/rfc5246>.

    [RFC5310] - Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R.,
         and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC
         5310, DOI 10.17487/RFC5310, February 2009, <http://www.rfc-
         editor.org/info/rfc5310>.

    [RFC5649] - Housley, R. and M. Dworkin, "Advanced Encryption Standard
         (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI
         10.17487/RFC5649, September 2009, <http://www.rfc-
         editor.org/info/rfc5649>.

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

    [RFC6325] - Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A.
         Ghanwani, "Routing Bridges (RBridges): Base Protocol
         Specification", RFC 6325, DOI 10.17487/RFC6325, July 2011,
         <http://www.rfc-editor.org/info/rfc6325>.

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

    [RFC7172] - Eastlake 3rd, D., Zhang, M., Agarwal, P., Perlman, R.,
         and D. Dutt, "Transparent Interconnection of Lots of Links
         (TRILL): Fine-Grained Labeling", RFC 7172, DOI
         10.17487/RFC7172, May 2014, <http://www.rfc-
         editor.org/info/rfc7172>.

    [RFC7176] - Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt,
         D., and A. Banerjee, "Transparent Interconnection of Lots of
         Links (TRILL) Use of IS-IS", RFC 7176, May 2014,
         <http://www.rfc-editor.org/info/rfc7176>.

    [RFC7177] - Eastlake 3rd, D., Perlman, R., Ghanwani, A., Yang, H.,
         and V. Manral, "Transparent Interconnection of Lots of Links

                   (TRILL): Adjacency", RFC 7177, DOI 10.17487/RFC7177, May 2014,
                   <http://www.rfc-editor.org/info/rfc7177>.

     [RFC7178] - Eastlake 3rd, D., Manral, V., Li, Y., Aldrin, S., and D.
                   Ward, "Transparent Interconnection of Lots of Links (TRILL):
                   RBridge Channel Support", RFC 7178, DOI 10.17487/RFC7178, May
                   2014, <http://www.rfc-editor.org/info/rfc7178>.

     [RFC7780] - Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A.,
                   Ghanwani, A., and S. Gupta, "Transparent Interconnection of
                   Lots of Links (TRILL): Clarifications, Corrections, and
                   Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016,
                   <http://www.rfc-editor.org/info/rfc7780>.

     [RFC7978] - Eastlake 3rd, D., Umair, M., and Y. Li, "Transparent
                   Interconnection of Lots of Links (TRILL): RBridge Channel
                   Header Extension", RFC 7978, DOI 10.17487/RFC7978, September
                   2016, <http://www.rfc-editor.org/info/rfc7978>.

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

     [TRILLoverIP] - M. Cullen, D. Eastlake, M. Zhang, D. Zhang,
                   "Transparent Interconnection of Lots of Links (TRILL) over IP",
                   draft-ietf-trill-over-ip, work in progress.

Informative References

     [RFC6234] - Eastlake 3rd, D. and T. Hansen, "US Secure Hash
                   Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI
                   10.17487/RFC6234, May 2011, <http://www.rfc-
                   editor.org/info/rfc6234>.

     [RFC7457] - Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing
                   Known Attacks on Transport Layer Security (TLS) and Datagram
                   TLS (DTLS)", RFC 7457, February 2015, <http://www.rfc-
                   editor.org/info/rfc7457>.

Acknowledgements

    The contributions of the following are hereby gratefully
    acknowledged:

        TBD

    The document was prepared in raw nroff. All macros used were defined
    within the source file.

Authors' Addresses

        Donald E. Eastlake, 3rd
        Huawei Technologies
        155 Beaver Street
        Milford, MA 01757 USA

        Phone: +1-508-333-2270
        EMail: d3e3e3@gmail.com

        Dacheng Zhang
        Huawei Technologies

        Email: dacheng.zhang@huawei.com

Copyright, Disclaimer, and Additional IPR Provisions

                 Simple Group Keying Protocol (SGKP)
                 <draft-ietf-trill-group-keying-04.txt>

Abstract

   This document specifies a simple general group keying protocol that
   provides for the distribution of shared secret keys to group members
   and the management of such keys. It assumes that secure pairwise keys
   can be created between any two group members.

Status of This Memo

Table of Contents

1. Introduction

   This document specifies a simple general group keying protocol that
   provides for the distribution of shared secret keys to group members
   and the management of such keys. It assumes that secure pairwise keys
   can be created between any two group members.

   A companion document specifies two profiles for the use of this group
   keying protocol in a case using DTLS and a case using IPsec payload
   formats. It is anticipated that there will be other uses for this
   group keying protocol.

1.1 Terminology and Acronyms

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119] [RFC8174]
   when, and only when, they appear in all capitals, as shown here.

   This document uses the following terminology and acronyms:

      AES - Advanced Encryption Standard.

      DTLS - Datagram Transport Level Security [RFC6347].

      GKd - A distinguished station in a group that is in charge of
         which group keying (Section 2) is in use.

      GKs - Stations in a group other than GKd (Section 2).

      IS-IS - Intermediate System to Intermediate System [RFC7176].

      keying material - The set of a Key ID, a secret key, and a cypher
         suite.

      QoS - Quality of Service.

      RBridge - An alternative term for a TRILL switch.

      TRILL - Transparent Interconnection of Lots of Links or Tunneled
         Routing in the Link Layer.

      TRILL switch - A device that implements the TRILL protocol
         [RFC6325] [RFC7780], sometimes referred to as an RBridge.

2. Simple Group Keying Protocol

   This section gives an overview of the assumptions and capabilities of
   the Simple Group Keying Protocol (SGKP) that provides shared secret
   group keys. Further details of the messages used for this protocol
   are give in Section 3.

   Any particular use of this protocol will require profiling giving
   further details and specifics for that use. For example, the envelope
   used for addressing and transmitting the messages of this protocol
   must be specified for any particular use. This protocol is not
   suitable for discovery messages but is intended for use between
   members of a group that have already established or can establish
   pair-wise security.

2.1 Assumptions

   The following are assumed:
      - All pairs of stations in the group can engage in pairwise
        communication with unicast messages and each can groupcast a
        message to the other group members.
      - At any particular time, there is a distinguished station GKd in
        the group that is in charge of keying for the groupcast data
        messages to be sent to the group. The group wide shared secret
        keys established by GKd are referred to herein as "dynamic"
        keys.
      - Pairwise keying has been negotiated between GKd and each other
        station GKs1, GKs2, ... GKsN in the group. These keys are
        referred to in this protocol as "pairwise" keys.
      - There are one or more keys, other than the dynamic or pairwise
        keys, which are already in place at all group member stations
        and may be present at other stations. These are referred to as
        "stable" keys.

   When keying material is stored by a station, it is accompanied by a
   "use flag" indicating whether or not that keying material is usable
   for groupcast transmissions.

2.2 Group Keying Procedure Overview

   GKd sends unicast keying messages to the other stations in the group
   and they respond as specified below and in further detail in the
   particular use profiles of SGKP. All such keying messages MUST be
   encrypted and authenticated using the pairwise keys as further
   specified in the use profile.

Typically, GKd sends a keying message to each GKs with keying
material. After successful acknowledgement of receipt from each GKs,
GKd sends a keying message to each GKs instructing it to use the
dynamic key GKd has set. It would be common for GKd to set a new
dynamic key at each GKs while an older dynamic key is in use so that
GKd can more promptly roll over to the new dynamic key when
appropriate.

To avoid an indefinite build up of keying material at a GKs, keys
have a lifetime specified by GKd and GKd can send a message deleting
a key. (GKd can also send a message indicating that a key is no
longer to be used but leaving it set.) Should the space available at
a GKs for keying material be exhausted, on receipt of a Set Key
keying message from GKd for a new key ID, GKs discards a dynamic key
it has and originates a Delete Key message to the source of that
dynamic key.

2.3 Transmission and Receipt of Group Data Messages

If a group has only one member, transmission of data between group
members is a moot question and any messages that would be so
transmitted if the group had more members are discarded.

If a group has only two members, then pairwise security is used
between them.

When a group has more than two members and a station in the group
transmits a data message to the group, if the transmitter has one or
more keys set by GKd that it has been instructed to use, it uses one
of those keys and its associated cypher suite to groupcast the data
message. If it has no such key, then it uses serial unicast to send
the data message to each other member of the group, negotiating
pairwise keys with them if it does not already have such pairwise
keys. Thus it is a responsibility of GKd not to authorize the use of
a groupcast key until it knows that all the GKs have that key.

When a station in the group receives data that has been groupcast to
the group, if the receiver has the key referenced by the data message
the receiver decrypts and verifies it. If verification fails or if
the receiver does not have the required key, the receiver discards
the data message. Thus whether GKs has been directed to "use" a key
by GKd is relevant only to transmission, not reception.

2.4 Changes in Group Membership or GKd

   When a new station joins the group, GKd SHOULD send that station the
   currently in-use group key and instruct it to use that key and MAY
   send it other keys known to the group members and intended for future
   use.

   If GKd detects that one or more stations that were members of the
   group are no longer members of the group, it SHOULD generate and
   distribute a new group key to the remaining group members, instruct
   them to use this new key, and delete from them any old keys known to
   the departed group member station(s) or at least instructing them to
   dis-use such old keys that are marked for use; however, in the case
   of groups with large and/or highly dynamic membership, where a
   station might frequently leave and then rejoin, it may, as a
   practical matter, be necessary to rekey less frequently.

   A new group member can become GKd due to the previous GKd leaving the
   group or a configuration change or the like. A GKs MUST NOT use
   keying material for transmission that was set by a station that it
   determines is not GKd. To avoid a gap in service, a station that is
   not GKd MAY set keying material at other stations in the group;
   however, such a non-GKd station cannot set the use flag for any such
   keying material. It is RECOMENDED that the second highest priority
   station to be GKd set such keying material at all other stations in
   the group. Should a station run out of room for keying material, it
   SHOULD discard keying material set by a station with lower priority
   to be GKd before discarding keying material set by a higher priority
   station and among keys set by GKd is SHOULD discard the lest recently
   used first.

3. Group Keying Messages

   Keying messages start with a Version number. This document specifies
   Version zero.

   Keying messages are structured, as shown in Figure 3.1 below, as
        o   a Version number,
        o   a Response flag,
        o   a Key ID length,
        o   the Key ID of a stable key,
        o   a group keying use profile identifier,
        o   possible padding,
        o   a key wrap algorithm specifier, and finally
        o   a key wrapped vector of additional fields wrapped using a key
            derived from the stable key identified.

   Keying messages are always sent unicast and encrypted and
   authenticated with the appropriate pairwise key, all as further
   specified for the particular use profile. It will typically be
   possible for GKd to calculate the keying message once, including the
   wrapping under a key derived from the stable key, then send that
   message to various GKs using the different pairwise keys for each
   GKs.

```
    +-+-+-+-+-+-+-+-+
    |Ver|R|KeyID1Lng|                 1 byte
    +-+-+-+-+-+-+-+-+...
    | KeyID1  ...                     KeyID1Lngth bytes
    +-+-+-+-+-+-+-+-+...
    | Use Type      |                 1 byte
    +-+-+-+-+-+-+-+-+
    | Pad1 Length   |                 1 byte
    +-+-+-+-+-+-+-+-+...
    | Padding                         Pad1 Length bytes
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
    | KW Al | KW Length        |       2 bytes
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
    |
    | Key Wrapped Material            Variable size
    |
    +-+-+-+-+-+-+-...
```
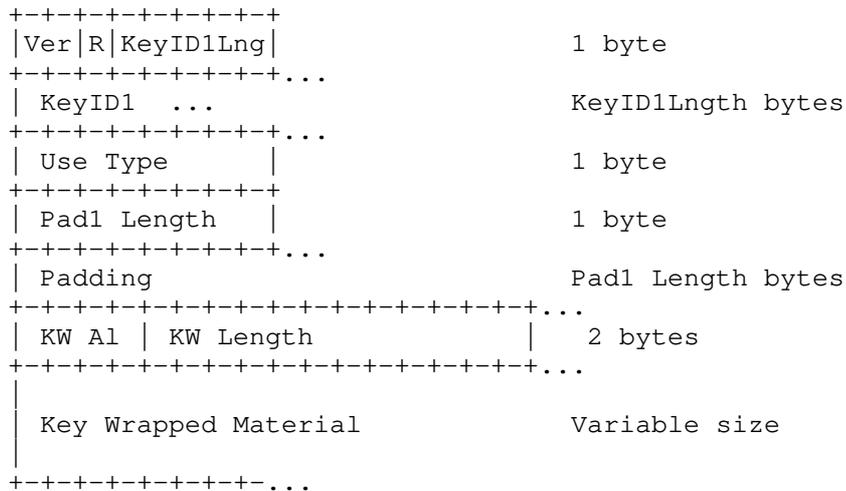
                    Figure 3.1. Keying Message Structure

   The fields in Figure 3.1 are as follows:

       Ver - Group Keying protocol version. This document specifies
          version zero.

       R - Response flag. If set to one, indicates a response message.

If set to zero, indicated a request or no-op message.

KeyID1Lngth, KeyID1 - KeyID1 identifies the stable key wrapping
    key (also known as the Key Encrypting Key (KEK)) as further
    specified in the use profile. KeyID1Lngth is a 5-bit field
    that gives the length of KeyID1 in bytes minus 1 as an
    unsigned integer.

Use Type - Specifies the particular group security use profile
    such as one of the two profiles in [SGKPuses]. See Section
    5, Item 3.

Pad1 Length, Pad1 - Padding to obscure the non-padded message
    size. Pad1 Length may be from 0 to 255 and gives the length
    of the padding as an unsigned integer. Each byte of padding
    MUST be equal to Pad1 Length. For example, 3 bytes of
    padding with length is 0x03030303.

KW Algorithm - An unsigned integer 4-bit field specifying the
    Key Wrap Algorithm. See Section 5, Item 4.

KW Length - An unsigned integer 14-bit field that gives the
    length of the Key Wrapped Material in octets.

Key Wrapped Material - The output of the designated Key
    Wrapping Algorithm on the message vector of fields using
    the designated stable key.

The vector of fields contained within the key wrapping is specified
for the various keying messages in subsections below. The contents
of this wrapped vector are protected by the key wrapping as well as
being authenticated and super-encrypted by the pairwise keyed
security used for sending the overall keying message. The probability
that the stable key used for key wrapping is the same as the outer
message pairwise key MUST be insignificant (less that 1 in 2**64).

Each group keying message contains, in the key wrapped vector of
fields, a message type and a message ID set by the sender of a
request. These fields are returned in the corresponding response to
assist in the matching of response to requests, except that there is
no response required to the No-Op message.

If no response is received to a request (other than a No-Op request)
for an amount of time configurable in milliseconds from 1 to (2**15
- 1), the request is re-transmitted with the same message ID. These
retries can occur up to a configurable number of times from 1 to 8.
Unless otherwise provided in the particular use profile, the default
response delay threshold is 200 milliseconds and the default maximum
number of retries is 3.

   Keying messages are sent with a priority/QoS configurable on a per
   device per use type basis. The default priority/QoS is specified in
   the use profile.

   Since the minimum length of the Key Wrapped Material is 16 bytes, the
   minimum valid length of a keying message before pairwise security is
   21 bytes, even if KeyID1 Length and Pad1 Length are zero. All multi-
   byte fields are in network order, that is, with the most significant
   byte first. The maximum valid length before pairwise security is 4
   (fixed bytes) + 32 (max KeyID1) + 255 (max padding) + 264 (max KW
   material) = 555 bytes.

3.1 Set Key Message

   The structure of the wrapped vector of fields for the Set Key keying
   message is as show in Figure 3.2. A recipient automatically
   determines the overall length provided for this vector of fields
   inside the key wrapping as a byproduct of the process of key
   unwrapping.

```
      +-+-+-+-+-+-+-+-+
      | Msg Type = 1  |                  1 bytes
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Msg ID                          3 bytes     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Pad2 Length   |                  1 bytes
      +-+-+-+-+-+-+-+-+...
      | Padding                         Pad2 Length bytes
      +-+-+-+-+-+-+-+-+...
      | Other                           Variable size
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Lifetime                    |   2 bytes
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | KeyID2 Length |                  1 byte
      +-+-+-+-+-+-+-+-+
      | KeyID2  ...                     KeyID2 Length bytes
      +-+-+-+-+-+-+-+-+
      | CypherSuiteLng|                  1 byte
      +-+-+-+-+-+-+-+-+
      | CypherSuite ...                 CypherSuiteLng bytes
      +-+-+-+-+-+-+-+-...
      | Key  ...                        Variable size
      +-+-+-+-+-+-+-+-...
```
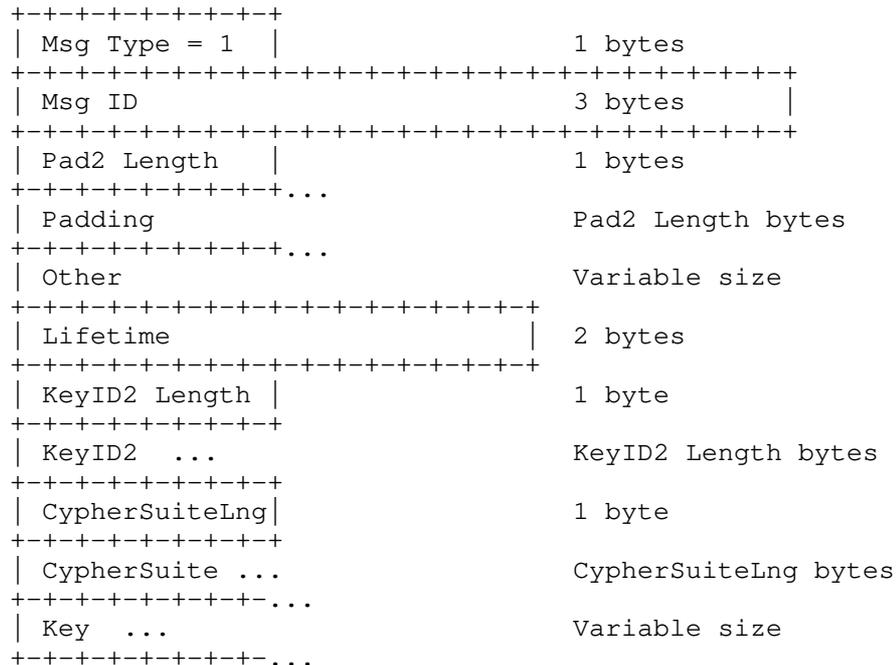
             Figure 3.2. Set Key Message Inner Structure

The fields are as follows:

Msg Type = 1 for Set Key message

Msg ID - A 3 byte quantity to be included in the corresponding
response message to assist in matching requests and
responses. Msg ID zero has a special meaning in responses
and MUST NOT be used in a Set Key message or any other
group keying request message.

Pad2 Length, Pad2 - Padding to obscure the size of the unpadded
AES wrapped data. Pad2 Length may be from 0 to 255 and
gives the length of the padding as an unsigned integer.
Each byte of padding MUST be equal to Pad1 Length. For
example, 2 bytes of padding with length byte is 0x020202.

Other - Additional information if specified in the use profile.
If Other information in this message is not mentioned in
the use profile, there is none and this portion of the
wrapped information is null. If a use profile specifies
Other information it must be possible to determine its
length so that following fields can be properly parsed and
so that the size of the Key field can be deduced; for
example, Other could begin with a length byte.

Lifetime - A 2-byte unsigned integer. After that number of
seconds plus one second, the key and associated information
being set MUST be discarded. Unless otherwise specified for
a particular use profile of this group keying protocol, the
default Lifetime is 15,000 seconds or a little over four
hours.

KeyID2 Length, KeyID2 - KeyID2 identifies the group key and
associated information being set as further specified in
the use profile. KeyID2 Length is an unsigned byte that
gives the length of KeyID2 in bytes.

CypherSuiteLng, CypherSuite - CypherSuite identifies the cypher
suite associated with the key being set as further
specified in the use profile. CypherSuite Length is an
unsigned byte the gives the length of CypherSuite in bytes.

Key - This is the actually group shared secret keying material
being set. Its length is deduced from the overall length of
the vector of fields (found by the key unwrap operation)
and the length of the preceding fields.

Keying material and associated cypher suite are indexed under the Key
ID and the identity of the station that sent the information. This
identity is normally the address of that station as specified in the

use profile.

If GKs already has a dynamic key set under KeyID2, the key's value
and associated cypher suite are compared with those in the Set Key
messages. If they are the same, the only receiver action is to update
the Lifetime information associated with KeyID2 and send a Response
message. If they are different, the lifetime, cypher suite, and key
(and possibly Other material) are replaced, the use flag is cleared,
and a Response message sent.

3.2 Use, Delete, Disuse, or Deleted Key Messages

   The structure of the wrapped material for the Use Key, Delete Key,
   Disuse Key, and Deleted Key keying messages are the same as each
   other except for the message type and are shown in Figure 3.3

```
    +-+-+-+-+-+-+-+-+
    | Msg Type = t  |                       1 byte
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Msg ID                                3 bytes       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Pad2 Length   |                       1 bytes
    +-+-+-+-+-+-+-+-+...
    | Padding                               Pad2 Length bytes
    +-+-+-+-+-+-+-+-+...
    | Other                                 Variable size
    +-+-+-+-+-+-+-+-+...
    | KeyID2 Length |                       1 byte
    +-+-+-+-+-+-+-+-+
    | KeyID2  ...                           KeyID2 bytes
    +-+-+-+-+-+-+-+-+
```
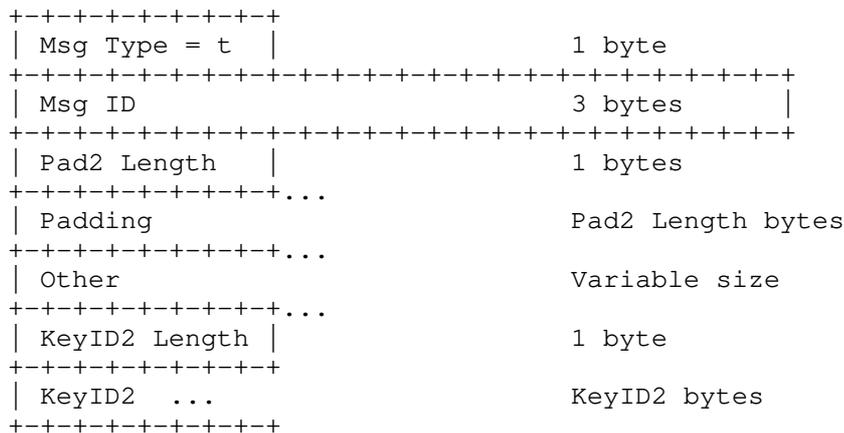
           Figure 3.3. Use, Delete, Disuse, or Deleted Key Message

   The Msg Type field specifies the particular message as follows:

```
   Msg Type    Message
   --------    ----------
        2      Use Key
        3      Delete Key
        4      Disuse Key
        5      Deleted Key
```

   The remaining fields are as specified in Section 3.1. KeyID2
   indicates the key to be used, deleted, for which use should cease, or
   which has been deleted, depending on the message type.

   It is RECOMMENDED that these messages be padded so as to be the same

length as a typical Set Key message.

The Delete Key is sent by a station believing itself to be GKd
instructing some GKs to delete a key. When a GKs spontaneously
deletes a key, it sends a Deleted Key message to the station from
which it received the key. The message types for Delete Key and
Deleted Key are different to minimize confusion in corner cases such
as the GKd changing while messages are in flight. The Msg ID used in
a Deleted Key message is created by the sending GKs from a space of
Msg IDs associated with that GKs, which space is independent of the
Msg IDs used in requests originated by GKd.

3.3 Response Message

The structure of the wrapped material for the Response group keying
message is as show below in Figure 3.4. A response message is
indicated by the R bit in the first byte of the message outside the
key wrapping.

A response MUST NOT be sent due to the receipt of a response. The R
bit is outside of the key wrapping so that this rule can be enforced
even in cases of difficulty in unwrapping.

```
+-+-+-+-+-+-+-+-+
| Msg Type = n  |                 1 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Msg ID                          3 bytes     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Pad2 Length   |                 1 bytes
+-+-+-+-+-+-+-+-+...
| Padding                         Pad2 Length bytes
+-+-+-+-+-+-+-+-+...
| Other                           Variable size
+-+-+-+-+-+-+-+-+...
| Response Code |                 1 byte
+-+-+-+-+-+-+-+-+
| ReqPartLength |                 1 byte
+-+-+-+-+-+-+-+-+...
| Request Part                    ReqPartLength bytes
+-+-+-+-+-+-+-+-...
```
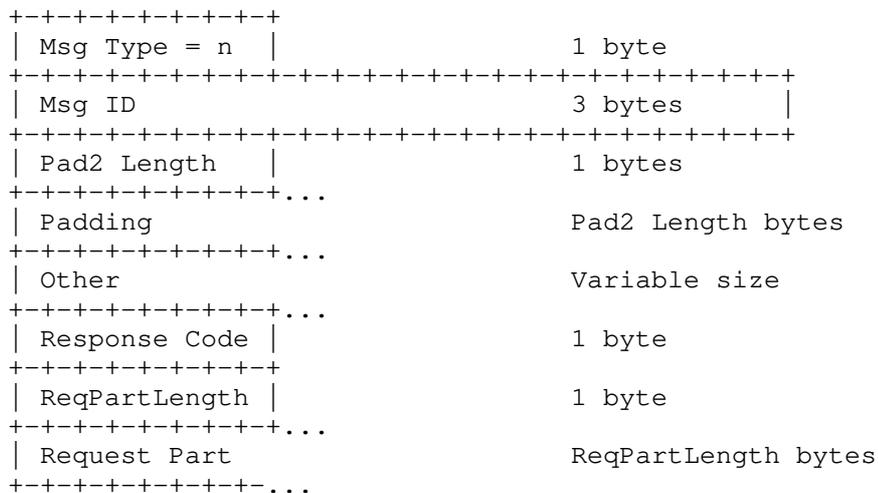
             Figure 3.4. Response Message Inner Structure

Except as specified below, the fields are as specified for the Key
Set message in Section 3.1.

   Msg Type, Msg ID - The content of these field is copied from
        the message in reply to which this Response message is sent

                    unless there is an error that stops the replying station
                    from determining them; in that case the special value zero
                    is used for the Msg Type and Msg ID. Errors where the Msg
                    Type and ID could not be determined are indicated by a
                    Response Code with its high order bit set to one, that is,
                    the 0b1xxxxxx bit set.

            Response Code - An unsigned byte giving the response as
                    enumerated in in Section 3.3.1. Any Response Code other
                    than a success indicates that the receiver took no action
                    on the request other than sending an error Response
                    message.

            ReqPartLength, Request Part: It is usually usefully to include
                    some or all of the request message in error responses.
                    - If the Response Code high order two bits are zero, the
                       request succeeded and ReqPartLength MUST be set to zero
                       so Request Part will be null.
                    - If the Response Code high order two bits are zero one
                       (0b01xxxxxx), then there was an error in the part of the
                       request inside the key wrapping but the unwrap process
                       was successful. ReqPartLength is the length of the
                       request message material included in the Request Part
                       field. The included request material is from the
                       unwrapped vector of fields started with the Msg Type
                       byte.
                    - If the Response Code high order bit is one (the
                       0b1xxxxxx is set), then there was an error parsing the
                       material outside the AES key wrap or an error in the AES
                       unwrapping process. ReqPartLength is the length of the
                       request message part included in the Request Part field.
                       The included part of the request starts with the first
                       byte of the message (the byte containing the version,
                       response flag, and KeyID1 Length). The key wrapped
                       material in the response message will still be wrapped.

   3.3.1 Response Codes

      The high order two bits of the Response Code have meaning as shown in
      Table 3.1.

         Top 2 Bits   Category
         ----------   ----------
            0b00   Success
            0b01   AES wrap contents
            0b10/11   Outside of AES wrap contents

                  Figure 3.1 Categories of Response Codes

```
   Response  Response
   Decimal      Hex   Meaning
   --------  --------  ----------
         0     0x00   Success
         1     0x01   Success and the key at an existing key ID was
                        changed
      2-47  0x02-0x2F  Unassigned
     48-63  0x30-0x3F  Reserved for special success codes defined in
                        use profiles
        64     0x40   Malformed inner fields (see Note 2 below)
        65     0x41   Unknown or zero Msg Type in a request
        66     0x42   Zero Msg ID in a request
        68     0x43   Invalid length KeyID2
        69     0x44   Unknown KeyID2
        70     0x45   Invalid length CypherSuite
        71     0x46   Unknown CyperSuite
        72     0x47   Bad Key (see Note 3 below)
    73-111  0x49-0x6F  Unassigned
   112-127  0x70-0x7F  Reserved for error codes defined in use
                        profiles and related to the key wrapped
                        contents
       128     0x80   Malformed message (see Note 1 below)
       129     0x81   Invalid length KeyID1
       130     0x82   Unknown KeyID1
       131     0x83   Unknown Use Type
       131     0x84   Key unwrap fails test for constant (e.g., AES
                        test 1, see Section 3 [RFC5649]).
       132     0x85   Key unwrap fails message length versus
                        wrapped size test (e.g., AES test 2,
                        see Section 3 [RFC5649]).
       133     0x86   Key unwrap fails test for value of padding
                        (e.g., AES test3, see Section 3
                        [RFC5649]).
   134-175  0x86-0x7F  Unassigned
   176-191  0xB0-0xBF  Reserved for error codes defined in use
                        profiles and related to parts of
                        message outside the key wrap contents
       192     0xC0   No keys set
       193     0xC1   Referenced key unknown
       194     0xC2   Referenced key known but use flag not set
   195-255  0xC3-0xFF  Reserved
```

Response Code Notes:

   Note 1 Message is too short or too long, AES wrapped material is too
           short, Padding bytes are not the required value, or similar
           fundamental message format problems.

   Note 2 The key wrapped inner vector of fields is too short or too
           long, Padding bytes are not the required value, or similar

fundamental vector of fields format problems.

Note 3   Key is not a valid length for CypherSuite or other internal
         checks on key (for example, parity bits in a 64 bit DES key
         (not that you should be using DES)) fail when they should be
         correct.

                     Figure 3.2 Response Codes

3.4 No-Op Message

   The No-Op message is a dummy message intended for use in disguising
   metadata deducible from keying message transmissions. It requires no
   response although a recipient can always decided to send a No-Op
   message to a station from which it has received such a message. The
   vector of fields inside the AES key wrap is as follows:

```
   +-+-+-+-+-+-+-+-+
   | Msg Type = 6  |                 1 byte
   +-+-+-+-+-+-+-+-+
   | Pad2 Length   |                 1 bytes
   +-+-+-+-+-+-+-+-+...
   | Padding                         Pad2 Length bytes
   +-+-+-+-+-+-+-+-...
```

                Figure 3.5. No-Op Message Inner Structure

   The Msg Type is set to 6 to indicate a No-Op message.

   Pad2 Length and Padding are as specified in Section 3.1. It is
   RECOMMENDED that Pad2 Length in a No-Op message be such as to make
   its length the same as the length of a typical Set Key message.

4. Security Considerations

   This section gives some general security considerations of this group
   keying protocol as distinguished from security considerations of a
   particular use profile.

   The method by which the stations in the group discover each other is
   specified in the group keying use profile. GKd controls group access
   and generally learns whatever it needs to know about GKs during the
   pairwise authentication and pairwise keying process.

   The group keying provided by this protocol is shared secret keying.
   This means that data messages can only be authenticated as coming
   from some group member but not as coming from a specific group
   member. If this level of authentication is insufficient, GKd can
   simply not set keys or not set them as usable. This will force all
   stations in the group that are configured to use security for multi-
   destination transmissions to the group to serially unicast data to
   the other group members using pairwise keying.

   The content value of padding fields in the Group Keying protocol is
   fixed so that it cannot be used as a covert channel. It might still
   be possible to use the length of padding as a covert channel.

5. IANA Considerations

    IANA is requested to perform the following actions:

       1. Establish a protocol parameters web page for "Group Keying
          Protocol Parameters" with the initial registries on that page
          as specified below in this section.

       2. Establish a "Message Type" registry on the Group Keying
          Protocol Parameters page as follows:

          Name: Message Types Registration Procedure: IETF Review
          Reference: [this document]

Type	Description	Reference
0	Reserved	[This document]
1	Set Key	[This document]
2	Use Key	[This document]
3	Delete Key	[This document]
4	Disuse Key	[This document]
5	Deleted Key	[This document]
6	No-Op	[This document]
7-250	Unassigned	
251-254	Reserved for Private Use	[This document]
255	Reserved	[This document]

       3. Establish a "Group Keying Use Profile" registry on the Group
          Keying Protocol Parameters page as follows:

          Name: Group Keying Use Profiles Registration Procedure: IETF
          Review Reference: [This document]

Profile	Description	Reference(s)
0	Reserved	[This document]
1	Extended RBridge Channel	[SGKPuses]
2	TRILL over IP	[SGKPuses]
3-250	Unassigned	
251-254	Reserved for Private Use	[This document]
255	Reserved	[This document]

       4. Establish a "Key Wrap Algorithm" registry on the Group Keying
          Protocol Parameters page as follows:

Name: Key Wrap Algorithms Registration Procedure: IETF Review
Reference: [This document]

```
    Code       Algorithm             Reference
    -----      -----------           ---------
      0        Reserved              [This document]
      1        AES                   [RFC5649]
      2        ChaCha                [ChaChaKW]
    3-16       -                     Unassigned
     17        Reserved              [This document]
```

5. Establish a "Response Code" registry on the Group Keying
   Protocol Parameters page as show below taking entries from the
   Response Code table in Section 3.3.1 above. In the table of
   values, the Reference column should be "[This document]" except
   where the Meaning is "Unassigned" or "Reserved".

   Name: Response Codes Registration Procedure: IETF Review
   Reference: [This document]

   Note: The top two bits of the Response Code indicate a category
   as specified in Section 3.3.1 of [this document].

```
    Response   Response
    Decimal      Hex      Meaning       Reference
    --------   ---------  -----------   ---------
        0        0x00     Success       [this document]
       ...        ...     ...
       255       0xFF     Reserved
```

Normative References

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

    [RFC5649] - Housley, R. and M. Dworkin, "Advanced Encryption Standard
          (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI
          10.17487/RFC5649, September 2009, <http://www.rfc-
          editor.org/info/rfc5649>.

    [RFC6325] - Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A.
          Ghanwani, "Routing Bridges (RBridges): Base Protocol
          Specification", RFC 6325, DOI 10.17487/RFC6325, July 2011,
          <http://www.rfc-editor.org/info/rfc6325>.

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

    [RFC7176] - Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt,
          D., and A. Banerjee, "Transparent Interconnection of Lots of
          Links (TRILL) Use of IS-IS", RFC 7176, May 2014,
          <http://www.rfc-editor.org/info/rfc7176>.

    [RFC7780] - Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A.,
          Ghanwani, A., and S. Gupta, "Transparent Interconnection of
          Lots of Links (TRILL): Clarifications, Corrections, and
          Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016,
          <http://www.rfc-editor.org/info/rfc7780>.

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

    [SGKPuses] - D. Eastlake, D. Zhang, "Simple Group Keying Protocol
          TRILL Use Profiles", draft-ietf-trill-link-gk-profiles, work in
          progress.

    [ChaChaKW] - D. Eastlake, "CHA CHA 20 Key Wrap with Padding
          Algorithm", draft-eastlake-chacha20-key-wrap, work in progress.

Informative References

        None.

Acknowledgements

   The contributions of the following are hereby gratefully
   acknowledged:

        TBD

Authors' Addresses

        Donald E. Eastlake, 3rd
        Futurewei Technologies
        1424 Pro Shop Court
        Davenport, FL 33896 USA

        Phone: +1-508-333-2270
        EMail: d3e3e3@gmail.com

        Dacheng Zhang
        Huawei Technologies

        Email: dacheng.zhang@huawei.com

TRILL WG   Radia Perlman
Internet-Draft   Dell EMC
Intended status: Standards Track   Fangwei Hu
Expires: September 12, 2018   ZTE Corporation
                                                      Donald Eastlake
                                                            Ting Liao
                                                  Huawei Technologies
                                                         Mar 11, 2018

                          TRILL Smart Endnodes
                  draft-ietf-trill-smart-endnodes-11.txt

   Abstract

      This draft addresses the problem of the size and freshness of the
      endnode learning table in edge RBridges, by allowing endnodes to
      volunteer for endnode learning and encapsulation/decapsulation. Such
      an endnode is known as a "Smart Endnode". Only the attached edge
      RBridge can distinguish a "Smart Endnode" from a "normal endnode".
      The Smart Endnode uses the nickname of the attached edge RBridge, so
      this solution does not consume extra nicknames. The solution also
      enables Fine Grained Label aware endnodes.

Table of Contents

1. Introduction

   The IETF TRILL (Transparent Interconnection of Lots of Links)
   protocol [RFC6325] [RFC7780] provides optimal pair-wise data frame
   forwarding without configuration, safe forwarding even during periods
   of temporary loops, and support for multipathing of both unicast and
   multicast traffic. TRILL accomplishes this by using IS-IS [IS-IS]
   [RFC7176] link state routing and encapsulating traffic using a header
   that includes a hop count. Devices that implement TRILL are called
   "RBridges" (Routing Bridges) or "TRILL Switches".

   An RBridge that attaches to endnodes is called an "edge RBridge" or
   "edge TRILL Switch", whereas one that exclusively forwards
   encapsulated frames is known as a "transit RBridge" or "transit TRILL
   Switch". An edge RBridge traditionally is the one that encapsulates
   a native Ethernet frame with a TRILL header, or that receives a
   TRILL-encapsulated packet and decapsulates the TRILL header. To

encapsulate efficiently, the edge RBridge must keep an "endnode
table" consisting of (MAC, Data Label, TRILL egress switch nickname)
sets, for those remote MAC addresses in Data Labels currently
communicating with endnodes to which the edge RBridge is attached.

These table entries might be configured, received from ESADI
[RFC7357], looked up in a directory [RFC7067], or learned from
decapsulating received traffic. If the edge RBridge has attached
endnodes communicating with many remote endnodes, this table could
become very large. Also, if one of the MAC addresses and Data Labels
in the table has moved to a different remote TRILL switch, it might
be difficult for the edge RBridge to notice this quickly, and because
the edge RBridge is encapsulating to the incorrect egress RBridge,
the traffic will get lost.

2. Conventions used in this document

2.1. Terminology

   Edge RBridge: An RBridge providing endnode service on at least one of
   its ports. It is also called an edge TRILL Switch.

   Data Label: VLAN or FGL.

   DRB: Designated RBridge [RFC6325].

   ESADI: End Station Address Distribution Information [RFC7357].

   FGL: Fine Grained Label [RFC7172].

   IS-IS: Intermediate System to Intermediate System [IS-IS].

   PDU: Protocol Data Unit.

   RBridge: Routing Bridge, an alternative name for a TRILL switch.

   Smart Endnode: An endnode that has the capability specified in this
   document including learning and maintaining (MAC, Data Label,
   Nickname) entries and encapsulating/decapsulating TRILL frame.

   Transit RBridge: An RBridge exclusively forwards encapsulated frames.
   It is also called a transit TRILL Switch.

   TRILL: Transparent Interconnection of Lots of Links
   [RFC6325][RFC7780].

   TRILL ES-IS: TRILL End System to Intermediate System, is a variation
   of TRILL IS-IS designed to operate on a TRILL link among and between
   one or more TRILL switches and end stations on that link[RFC8171].

   TRILL Switch: a device that implements the TRILL protocol; an
   alternative term for an RBridge.

2.2. Requirements Language

   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.

3. Solution Overview

   The Smart Endnode solution defined in this document addresses the
   problem of the size and freshness of the endnode learning table in
   edge RBridges. An endnode E, attached to an edge RBridge R, tells R
   that E would like to be a "Smart Endnode", which means that E will
   encapsulate and decapsulate the TRILL frame, using R's nickname.
   Because E uses R's nickname, this solution does not consume extra
   nicknames.

   Take Figure 1 as the example Smart Endnode scenario: RB1, RB2 and RB3
   are the RBridges in the TRILL domain, and SE1 and SE2 are the Smart
   Endnodes which can encapsulate and decapsulate the TRILL packets.
   RB1 is the edge RB that SE1 and SE2 have attached to. RB1 assigns
   one of its nicknames to be used by SE1 and SE2.

   Each Smart Endnode, SE1 and SE2, uses RB1's nickname when
   encapsulating, and maintains an endnode table of (MAC, label, TRILL
   egress switch nickname) for remote endnodes that it (SE1 or SE2) is
   corresponding with. RB1 does not decapsulate packets destined for
   SE1 or SE2, and does not learn (MAC, label, TRILL egress switch
   nickname) for endnodes corresponding with SE1 or SE2, but RB1 does
   decapsulate, and does learn (MAC, label, TRILL egress switch
   nickname) for any endnodes attached to RB1 that have not declared
   themselves to be Smart Endnodes.

   Just as an RBridge learns and times out (MAC, label, TRILL egress
   switch nickname), Smart Endnodes SE1 and SE2 also learn and time out
   endnode entries. However, SE1 and SE2 might also determine, through
   ICMP messages or other techniques that an endnode entry is not
   successfully reaching the destination endnode, and can be deleted,
   even if the entry has not timed out.

If SE1 wishes to correspond with destination MAC D, and no endnode
entry exists, SE1 will encapsulate the packet as an unknown
destination, or consulting a directory [RFC7067] (just as an RBridge
would do if there was no endnode entry).
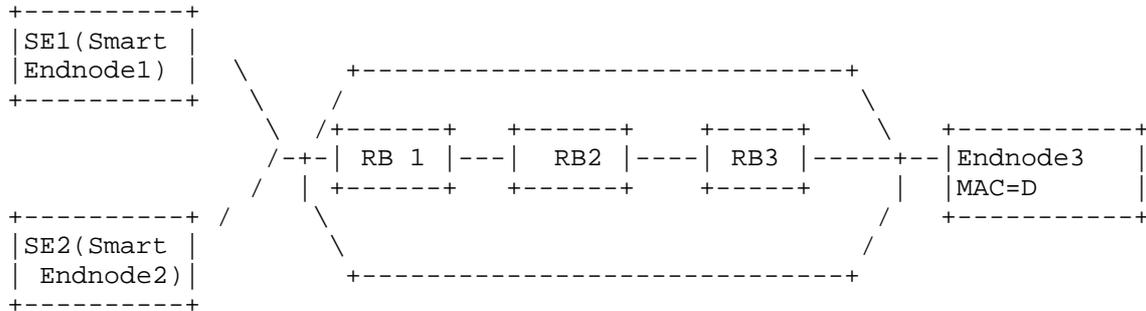
```
+----------+
|SE1(Smart |
|Endnode1) |  \          +----------------------------+
+----------+   \     /                                 \
                \   /+------+   +------+    +-----+    \  +----------+
                /-+-| RB 1 |---| RB2  |----| RB3 |-----+--|Endnode3  |
               /  | +------+   +------+    +-----+     |  |MAC=D     |
+----------+  /    \                                  /  +----------+
|SE2(Smart |   \    \                                /
| Endnode2)|    \        +----------------------------+
+----------+
```
Figure 1 Smart Endnode Scenario

The mechanism in this draft is that the Smart Endnode SE1 issues a
Smart-Hello, indicating SE1's desire to act as a Smart Endnode,
together with the set of MAC addresses and Data Labels that SE1 owns.
The Smart-Hello is used to announce the Smart Endnode capability and
parameters (such as MAC address, Data Label etc.). The Smart-Hello
is a type of TRILL ES-IS PDU, which is specified in section 5 of
[RFC8171]. The detailed content for a Smart Endnode's Smart-Hello is
defined in section 4.

If RB1 supports having a Smart Endnode neighbor it also sends Smart-
Hellos. The Smart Endnode learns from RB1's Smart-Hellos what RB1's
nickname is and which trees RB1 can use when RB1 ingresses multi-
destination frames. Although Smart Endnode SE1 transmits Smart-
Hellos, it does not transmit or receive LSPs or E-L1FS FS-LSPs
[RFC7780].

Since a Smart Endnode can encapsulate TRILL Data packets, it can
cause the Inner.Lable to be a Fine Grained Label [RFC7172], thus this
method supports FGL aware endnodes. When and how a Smart Endnode
decides to use the FGL instead of VLANs to encapsulate the TRILL Data
packet is out of scope in this document.

4. Smart-Hello Mechanism between Smart Endnode and RBridge

   The subsections below describe Smart-Hello messages.

4.1. Smart-Hello Encapsulation

   Although a Smart Endnode is not an RBridge, does not send LSPs or
   maintain a copy of the link state database, and does not perform
   routing calculations, it is required to have a "Hello" mechanism (1)
   to announce to edge RBridges that it is a Smart Endnode and (2) to
   tell them what MAC addresses it is handling in what Data Labels.
   Similarly, an edge RBridge that supports Smart Endnodes needs a
   message (1) to announce that support, (2) to inform Smart Endnodes
   what nickname to use for ingress and what nickname(s) can be used as
   egress nickname in a multi-destination TRILL Data packet, and (3) the
   list of Smart Endnodes it knows about on that link.

   The messages sent by Smart Endnodes and by edge RBridges that support
   Smart Endnodes are called "Smart-Hellos". The Smart-Hello is a type
   of TRILL ES-IS PDU, which is specified in [RFC8171].

   The Smart-Hello Payload, both for Smart-Hellos sent by Smart Endnodes
   and for Smart-Hellos sent by Edge RBridges, consists of TRILL IS-IS
   TLVs as described in the following two sub-sections. The non-
   extended format is used so TLVs, sub-TLVs, and APPsub-TLVs have an
   8-bit size and type field. Both types of Smart-Hellos MUST include a
   Smart-Parameters APPsub-TLV as follows inside a TRILL GENINFO TLV:

```
          +-+-+-+-+-+-+-+-+-
          |Smart-Parameters|              (1 byte)
          +-+-+-+-+-+-+-+-+-
          |    Length      |              (1 byte)
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
          |        Holding Time           | (2 bytes)
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
          |          Flags                | (2 bytes)
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
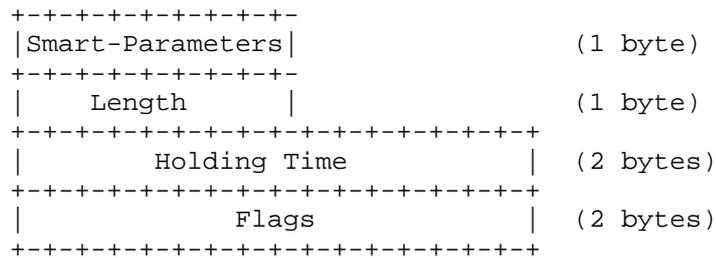
                  Figure 2 Smart Parameters APPsub-TLV

   o Type: APPsub-TLV type Smart-Parameters, value is TBD1.

   o Length: 4.

   o Holding Time: A time in seconds as an unsigned integer. It has the
   same meaning as the Holding Time field in IS-IS Hellos [IS-IS]. A
   Smart Endnode and an Edge RBridge supporting Smart Endnodes MUST send
   a Smart-Hello at least three times during their Holding Time. If no
   Smart-Hellos is received from a Smart Endnode or Edge RBridge within
   the most recent Holding Time it sent, it is assumed that it is no
   longer available.

o Flags: At this time all of the Flags are reserved and MUST be send
as zero and ignored on receipt.

If more than one Smart Parameters APPsub-TLV appears in a Smart-
Hello, the first one is used and any following ones are ignored. If
no Smart Parameters APPsub-TLV appears in a Smart-Hello, that Smart-
Hello is ignored.

4.2. Edge RBridge's Smart-Hello

The edge RBridge's Smart-Hello contains the following information in
addition to the Smart-Parameters APPsub-TLV:

o RBridge's nickname. The nickname sub-TLV, specified in section
2.3.2 in [RFC7176], is reused here carried inside a TLV 242 (IS-IS
router capability) in a Smart-Hello frame. If more than one nickname
appears in the Smart-Hello, the first one is used and the following
ones are ignored.

o Trees that RB1 can use when ingressing multi-destination frames.
The Tree Identifiers Sub-TLV, specified in section 2.3.4 in
[RFC7176], is reused here.

o Smart Endnode neighbor list. The TRILL Neighbor TLV, specified in
section 2.5 in [RFC7176], is reused for this purpose.

An Authentication TLV MAY also be included.

4.3. Smart Endnode's Smart-Hello

A new APPsub-TLV (Smart-MAC TLV) is defined for use by Smart Endnodes
as defined below. In addition, there will be a Smart-Parameters
APPsub-TLV and there MAY be an Authentication TLV in a Smart Endnode
Smart-Hello.

If there are several VLANs/FGL Data Labels for that Smart Endnode,
the Smart-MAC APPsub-TLV is included several times in Smart Endnode's
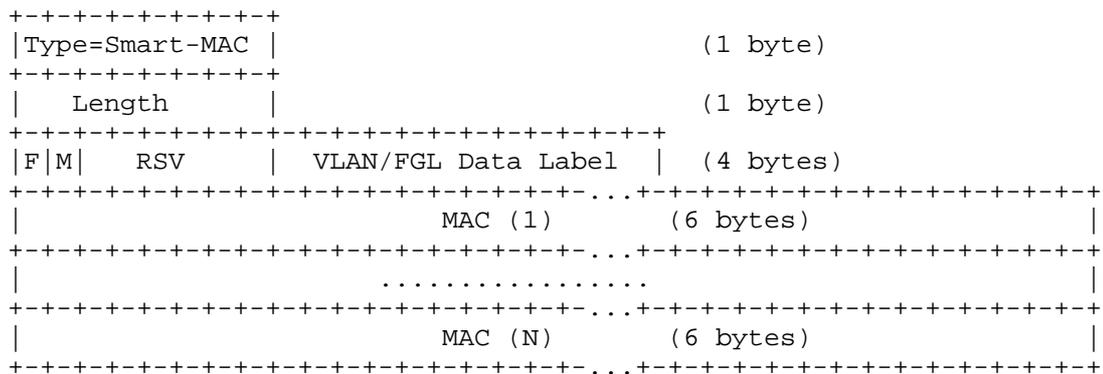Smart-Hello. This APPsub-TLV appears inside a TRILL GENINFO TLV.

```
+-+-+-+-+-+-+-+-+
|Type=Smart-MAC |                              (1 byte)
+-+-+-+-+-+-+-+-+
|   Length      |                              (1 byte)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|M|   RSV     | VLAN/FGL Data Label |        (4 bytes)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         MAC (1)        (6 bytes)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        ................                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         MAC (N)        (6 bytes)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Figure 3 Smart-MAC APPsub-TLV

   o Type: TRILL APPsub-TLV Type Smart-MAC, value is TBD2.

   o Length: Total number of bytes contained in the value field of the
   TLV, that is, the sum of the length of the F/M/RSV/FGL Data Label
   fields and 6 times the number of MAC addresses present. So, if there
   are n MAC addresses, this is 4+6*n.

   o F: 1 bit. If it is set to 1, it indicates that the endnode
   supports FGL data labels [RFC7172], and that this Smart-MAC APPsub-
   TLV has an FGL in the following VLAN/FGL field. Otherwise, the VLAN/
   FGL Data Label field is a VLAN ID.(See below for the format of the
   VLAN/FGL Data Label field).

   o M: 1 bit. If it is set to 1, it indicates multi-homing(See
   Section 6). If it is set to 0, it indicates that the Smart Endnodes
   are not using multi-homing.

   o RSV: 6 bits, is reserved for the future use.

   o VLAN/FGL Data Label: 24bits. If F is 1, this field is a 24-bit FGL
   Data Label for all subsequent MAC addresses in this APPsub-TLV.
   Otherwise, if F is 0, the lower 12 bits is the VLAN of all subsequent
   MAC addresses in this APPsub-TLV, and the upper 12 bits is not
   used(sent as zero and ignored on receipt). If there is no VLAN/FGL
   data label specified, the VLAN/FGL Data Label is zero.

   o MAC(i): This is a 48-bit MAC address reachable in the Data Label
   sent by the Smart Endnode that is announcing this APPsub-TLV.

5. Data Packet Processing

   The subsections below specify Smart Endnode data packet processing.
   All TRILL Data packets sent to or from Smart Endnodes are sent in the
   Designated VLAN [RFC6325] of the local link but do not necessarily
   have to be VLAN tagged.

5.1. Data Packet Processing for Smart Endnode

   A Smart Endnode does not issue or receive LSPs or E-L1FS FS-LSPs or
   calculate topology. It does the following:

   o A Smart Endnode maintains an endnode table of (the MAC address of
      remote endnode, Data Label, the nickname of the edge RBridge's
      attached) entries of end nodes with which the Smart Endnode is
      communicating. Entries in this table are populated the same way
      that an edge RBridge populates the entries in its table:

      * learning from (source MAC address ingress nickname) on packets
         it decapsulates.

      * by querying a directory [RFC7067].

      * by having some entries configured.

   o When Smart Endnode SE1 wishes to send unicast frame to remote node
      D, if (MAC address of remote endnode D, Data Label, nickname)
      entry is in SE1's endnode table, SE1 encapsulates the ingress
      nickname as the nickname of the RBridge(RB1), egress nickname as
      indicated in D's table entry. If D is unknown, SE1 either queries
      a directory or encapsulates the packet as a multi-destination
      frame, using one of the trees that RB1 has specified in RB1's
      Smart-Hello. The mechanism for querying a directory is given in
      [RFC8171].

   o When SE1 wishes to send a BUM packet to the TRILL campus, SE1
      encapsulates the packet using one of the trees that RB1 has
      specified.

   If the Smart Endnode SE1 sends a multi-destination TRILL Data packet,
   the destination MAC of the outer Ethernet is the All-RBridges
   multicast address.

   The Smart Endnode SE1 need not send Smart-Hellos as frequently as
   normal RBridges. These Smart-Hellos could be periodically unicast to
   the Appointed Forwarder RB1. In case RB1 crashes and restarts, or
   the DRB changes and SE1 receives the Smart-Hello without mentioning
   SE1, SE1 SHOULD send a Smart-Hello immediately. If RB1 is Appointed

Forwarder for any of the VLANs that SE1 claims, RB1 MUST list SE1 in
its Smart-Hellos as a Smart Endnode neighbor.

5.2. Data Packet Processing for Edge RBridge

The attached edge RBridge processes and forwards TRILL Data packets
based on the endnode property rather than for encapsulation and
forwarding the native frames the same way as the traditional
RBridges. There are several situations for the edge RBridges as
follows:

o If receiving an encapsulated unicast TRILL Data packet from a port
with a Smart Endnode, with RB1's nickname as ingress, the edge
RBridge RB1 forwards the frame to the specified egress nickname, as
with any encapsulated frame. However, RB1 SHOULD filter the
encapsulation frame based on the inner source MAC and Data Label as
specified for the Smart Endnode. If the MAC (or Data Label) are not
among the expected entries of the Smart Endnode, the frame would be
dropped by the edge RBridge. If the edge RBridge does not perform
this check, it makes it easier for a rogue end station to inject
bogus TRILL Data packets into the TRILL campus.

o If receiving a unicast TRILL Data packet with RB1's nickname as
egress from the TRILL campus, and the destination MAC address in the
enclosed packet is a MAC address that has been listed by a "Smart
Endnode", RB1 leaves the packet encapsulated to that Smart Endnode.
The outer Ethernet destination MAC is the destination Smart Endnode's
MAC address, the inner destination MAC address is either the Smart
Endnode's MAC address or some other MAC address that the Smart
Endnode advertised in its Smart Hello, and the outer Ethernet source
MAC address is the RB1's port MAC address. The edge RBridge still
decreases the Hop count value by 1, for there is one hop between the
RB1 and Smart Endnode.

o If receiving a multi-destination TRILL Data packet from a port with
a Smart Endnode, RBridge RB1 forwards the TRILL encapsulation to the
TRILL campus based on the distribution tree indicated by the egress
nickname. If the egress nickname does not correspond to a
distribution tree, the packet is discarded. If there are any normal
endnodes (i.e, non-Smart Endnodes) attached to the edge RBridge RB1,
RB1 decapsulates the frame and sends the native frame to these ports
possibly pruned based on multicast listeners, in addition to
forwarding the multi-destination TRILL frame to the rest of the
campus.

o If RB1 receives a native multi-destination data frame, which is
sent by a non-Smart Endnode, from a port, including hybrid endnodes
(Smart Endnodes and non-Smart Endnodes), RB1 will encapsulate it as

multi-destination TRILL Data packet , and send the encapsulated
multi-destination TRILL Data Packet out that same port to the Smart
Endnodes attached to the port, and also send the encapsulated multi-
destination TRILL Data Packet to the TRILL campus through other
ports.

o If RB1 receives a multi-destination TRILL Data packet from a remote
RBridge, and the exit port includes hybrid endnodes(Smart Endnodes
and non-Smart Endnodes), it sends two copies of multicast frames out
the port, one as native and the other as TRILL encapsulated frame.
When Smart Endnode receives multi-destination TRILL Data packet, it
learns the remote (MAC address, Data Label, Nickname) entry. A Smart
Endnodes ignores native data frames. A normal (non-Smart) Endnode
receives the native frame and learns the remote MAC address and
ignores the TRILL data packet. This transit solution may bring some
complexity for the edge RBridge and waste network bandwidth resource,
so avoiding the hybrid endnodes scenario by attaching the Smart
Endnodes and non-Smart Endnodes to different ports is RECOMMENDED.

6. Multi-homing Scenario

   Multi-homing is a common scenario for the Smart Endnode. The Smart
   Endnode is on a link attached to the TRILL domain in two places: to
   edge RBridge RB1 and RB2. Take the figure below as example. The
   Smart Endnode SE1 is attached to the TRILL domain by RB1 and RB2
   separately. Both RB1 and RB2 could announce their nicknames to SE1.

```
                      . ....................
                      . +------+            .
                      . | RB1  |            .
                      . /+------+            .
         +----------+ ./            +-----+ .   +----------+
         |SE1(Smart |/.             | RB3 |......| Smart    |
         | Endnode1)| .\            +-----+ .   | Endnode2 |
         +----------+ . \                   .   +----------+
                      . +-----+             .
                      . | RB2 |   TRILL     .
                      . +-----+   Domain    .
                      ......................
```
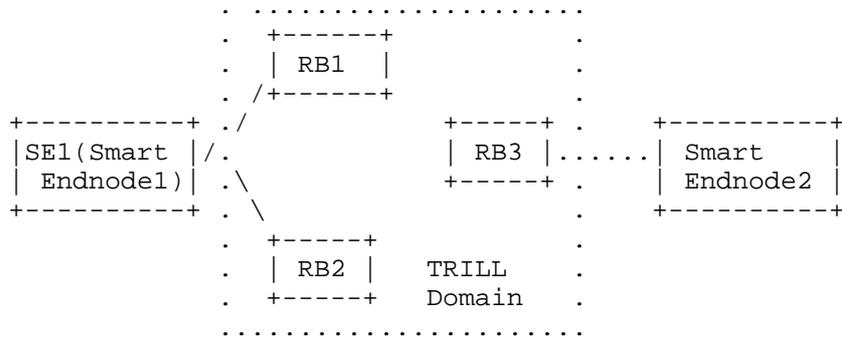
                    Figure 4 Multi-homing Scenario

   Smart Endnode SE1 can choose either RB1 or RB2's nickname, when
   encapsulating and forwarding a TRILL data packet. If the active-
   active load balance is considered for the multi-homing scenario, the
   Smart Endnode SE1 could use both RB1 and RB2's nickname to
   encapsulate and forward TRILL Data packet. SE1 uses RB1's nickname

when forwarding through RB1, and RB2's nickname when forwarding
through RB2. This will cause MAC flip-flopping(see [RFC7379]) of the
endnode table entry in the remote RBridges (or Smart Endnodes). The
solution for the MAC flip-flopping issue is to set a multi- homing
bit in the RSV field of the TRILL data packet. When remote RBridge
RB3 or Smart Endnodes receives a data packet with the multi-homed bit
set, the endnode entries (SE1's MAC address, label, RB1's nickname)
and (SE1's MAC address, label, RB2's nickname) will coexist as
endnode entries in the remote RBridge. (An alternative solution
would be to use the ESADI protocol to distribute multiple attachments
of a MAC address of a multi-homing group, The ESADI is deployed among
the edge RBridges (See section 5.3 of [RFC7357])).

7. Security Considerations

   Smart-Hellos can be secured by using Authentication TLVs based on
   [RFC5310]. If they are not secured, then it is easier for a rogue
   end station that does not posses the required keying material to be
   falsely recognized as a valid Smart Endnode.

   For general TRILL Security Considerations, see [RFC6325]. As stated
   there, since end stations are connected to edge RBridge ports by
   Ethernet, those ports MAY require end stations to authenticate
   themselves using [IEEE802.1X] and authenticate and encrypt traffic
   to/from the RBridge port with [IEEE802.1AE].

   If they misbehave, Smart Endnodes can forge arbitrary ingress and
   egress nicknames in the TRILL Headers of the TRILL Data packets they
   construct. Decapsulating at egress RBridges or remote Smart Endnodes
   that believe such a forged ingress nickname would send future traffic
   destined for the inner source MAC address of the TRILL Data frame to
   the wrong edge RBridge if data plane learning is in use. Because of
   this, an RBridge port should not be configured to support Smart
   Endnodes unless the end stations on that link are trusted or can be
   adequately authenticated.

   As with any end station, Smart Endnodes can forge the outer MAC
   addresses of packets they send (See Section 6 of [RFC6325].) Because
   they encapsulate TRILL Data packets, they can also forge inner MAC
   addresses. The encapsulation performed by Smart Endnodes also means
   they can send data in any Data Label which means they must be trusted
   in order to enforce a security policy based on Data Labels.

   The TRILL-Hello is a type of TRILL ES-IS, and is defined in
   [RFC8171]. Receiving and processing TRILL-Hello for RBridges and
   Smart Endnodes would not bring more security and vulnerability issues
   than the TRILL ES-IS security defined in [RFC8171].

For added security against the compromise of data due to its mis-
delivery for any reason, including the above, end-to-end encryption
and authentication should be considered; that is, encryption and
authentication from source end station to destination end station.

The mechanism described in this document requires Smart Endnodes to
be aware of the MAC address(es) of the TRILL edge RBridge(s) to which
they are attached and the egress RBridge nickname from which the
destination of the packets is reachable. With that information,
Smart Endnodes can learn a substantial amount about the topology of
the TRILL domain. Therefore, there could be a potential security
risk when the Smart Endnodes are not trusted or are compromised.

8. IANA Considerations

IANA is requested to allocate APPsub-TLV type numbers for the Smart-
MAC and Smart-Parameters APPsub-TLVs from the range below 256 and
update the "TRILL APPsub-TLV Types under IS-IS TLV 251 Application
Identifier 1" registry as follows.

```
+----------+-------------------+-----------------+
| Protocol |    Description    |    Reference    |
+----------+-------------------+-----------------+
|   TBD1   |  Smart-Parameters | [this document] |
|   TBD2   |     Smart-MAC     | [this document] |
+----------+-------------------+-----------------+
```

Table 1

9. Acknowledgements

The contributions of the following persons are gratefully
acknowledged: Mingui Zhang, Weiguo Hao, Linda Dunbar, Kesava Vijaya
Krupakaran and Andrew Qu.

10. References

10.1. Informative References

[IEEE802.1AE]
          "IEEE Standard for Local and metropolitan area networks--
          Media Access Control (MAC) Security.", 2006.

[IEEE802.1X]
          "IEEE Standard for Local and metropolitan area networks--
          Port-Based Network Access Control", 2010.

   [RFC7067] Dunbar, L., Eastlake 3rd, D., Perlman, R., and I.
              Gashinsky, "Directory Assistance Problem and High-Level
              Design Proposal", RFC 7067, DOI 10.17487/RFC7067, November
              2013, <https://www.rfc-editor.org/info/rfc7067>.

   [RFC7379] Li, Y., Hao, W., Perlman, R., Hudson, J., and H. Zhai,
              "Problem Statement and Goals for Active-Active Connection
              at the Transparent Interconnection of Lots of Links
              (TRILL) Edge", RFC 7379, DOI 10.17487/RFC7379, October
              2014, <https://www.rfc-editor.org/info/rfc7379>.

10.2. Normative References

   [IS-IS]   ISO/IEC 10589:2002, Second Edition,, "Intermediate System
              to Intermediate System Intra-Domain Routing Exchange
              Protocol for use in Conjunction with the Protocol for
              Providing the Connectionless-mode Network Service (ISO
              8473)", 2002.

   [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>.

   [RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R.,
              and M. Fanto, "IS-IS Generic Cryptographic
              Authentication", RFC 5310, DOI 10.17487/RFC5310, February
              2009, <https://www.rfc-editor.org/info/rfc5310>.

   [RFC6325] Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A.
              Ghanwani, "Routing Bridges (RBridges): Base Protocol
              Specification", RFC 6325, DOI 10.17487/RFC6325, July 2011,
              <https://www.rfc-editor.org/info/rfc6325>.

   [RFC7172] Eastlake 3rd, D., Zhang, M., Agarwal, P., Perlman, R., and
              D. Dutt, "Transparent Interconnection of Lots of Links
              (TRILL): Fine-Grained Labeling", RFC 7172,
              DOI 10.17487/RFC7172, May 2014,
              <https://www.rfc-editor.org/info/rfc7172>.

   [RFC7176] Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt,
              D., and A. Banerjee, "Transparent Interconnection of Lots
              of Links (TRILL) Use of IS-IS", RFC 7176,
              DOI 10.17487/RFC7176, May 2014,
              <https://www.rfc-editor.org/info/rfc7176>.

   [RFC7357] Zhai, H., Hu, F., Perlman, R., Eastlake 3rd, D., and O.
              Stokes, "Transparent Interconnection of Lots of Links
              (TRILL): End Station Address Distribution Information
              (ESADI) Protocol", RFC 7357, DOI 10.17487/RFC7357,
              September 2014, <https://www.rfc-editor.org/info/rfc7357>.

   [RFC7780] Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A.,
              Ghanwani, A., and S. Gupta, "Transparent Interconnection
              of Lots of Links (TRILL): Clarifications, Corrections, and
              Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016,
              <https://www.rfc-editor.org/info/rfc7780>.

   [RFC8171] Eastlake 3rd, D., Dunbar, L., Perlman, R., and Y. Li,
              "Transparent Interconnection of Lots of Links (TRILL):
              Edge Directory Assistance Mechanisms", RFC 8171,
              DOI 10.17487/RFC8171, June 2017,
              <https://www.rfc-editor.org/info/rfc8171>.

   [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>.

Authors' Addresses

   Radia Perlman
   Dell EMC
   176 South Street
   Hopkinton, MA 01748
   USA

   Phone: +1-206-291-367
   Email: radiaperlman@gmail.com

   Fangwei Hu
   ZTE Corporation
   No.889 Bibo Rd
   Shanghai 201203
   China

   Phone: +86 21 68896273
   Email: hu.fangwei@zte.com.cn

Donald Eastlake
Huawei Technologies
155 Beaver Street
Milford, MA 01757
USA

Phone: +1-508-634-2066
Email: d3e3e3@gmail.com

Ting Liao
Huawei Technologies
Nanjing, Jiangsu 210012
China

Email: liaoting1@huawei.com