Stream: Internet Engineering Task Force (IETF)
RFC: 9185
Category: Informational
Published: April 2022
ISSN: 2070-1721
Authors: P. Jones, *Cisco Systems*; P. Ellenbogen, *Princeton University*; N. Ohlmeier, *8x8, Inc.*

RFC 9185
DTLS Tunnel between a Media Distributor and Key Distributor to Facilitate Key Exchange

Abstract

This document defines a protocol for tunneling DTLS traffic in multimedia conferences that enables a Media Distributor to facilitate key exchange between an endpoint in a conference and the Key Distributor. The protocol is designed to ensure that the keying material used for hop-by-hop encryption and authentication is accessible to the Media Distributor, while the keying material used for end-to-end encryption and authentication is inaccessible to the Media Distributor.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9185.

Copyright Notice

Table of Contents

1. Introduction

An objective of Privacy-Enhanced RTP Conferencing (PERC) [RFC8871] is to ensure that endpoints in a multimedia conference have access to the end-to-end (E2E) and hop-by-hop (HBH) keying material used to encrypt and authenticate Real-time Transport Protocol (RTP) packets [RFC3550], while the Media Distributor has access only to the HBH keying material for encryption and authentication.

This specification defines a tunneling protocol that enables the Media Distributor to tunnel DTLS messages [RFC9147] between an endpoint and a Key Distributor, thus allowing an endpoint to use DTLS for the Secure Real-time Transport Protocol (DTLS-SRTP) [RFC5764] for establishing encryption and authentication keys with the Key Distributor.

The tunnel established between the Media Distributor and Key Distributor is a TLS connection [RFC8446] that is established before any messages are forwarded by the Media Distributor on behalf of endpoints. DTLS packets received from an endpoint are encapsulated by the Media Distributor inside this tunnel as data to be sent to the Key Distributor. Likewise, when the Media Distributor receives data from the Key Distributor over the tunnel, it extracts the DTLS message inside and forwards the DTLS message to the endpoint. In this way, the DTLS association for the DTLS-SRTP procedures is established between an endpoint and the Key Distributor, with the Media Distributor forwarding DTLS messages between the two entities via the established tunnel to the Key Distributor and having no visibility into the confidential information exchanged.

Following the existing DTLS-SRTP procedures, the endpoint and Key Distributor will arrive at a selected cipher and keying material, which are used for HBH encryption and authentication by both the endpoint and the Media Distributor. However, since the Media Distributor would not have direct access to this information, the Key Distributor explicitly shares the HBH key information with the Media Distributor via the tunneling protocol defined in this document. Additionally, the endpoint and Key Distributor will agree on a cipher for E2E encryption and authentication. The Key Distributor will transmit keying material to the endpoint for E2E operations but will not share that information with the Media Distributor.

By establishing this TLS tunnel between the Media Distributor and Key Distributor and implementing the protocol defined in this document, it is possible for the Media Distributor to facilitate the establishment of a secure DTLS association between an endpoint and the Key Distributor in order for the endpoint to generate E2E and HBH keying material. At the same time, the Key Distributor can securely provide the HBH keying material to the Media Distributor.

2. Conventions Used in This Document

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.

This document uses the terms "endpoint", "Media Distributor", and "Key Distributor" defined in [RFC8871].

3. Tunneling Concept

A TLS connection (tunnel) is established between the Media Distributor and the Key Distributor. This tunnel is used to relay DTLS messages between the endpoint and Key Distributor, as depicted in Figure 1:

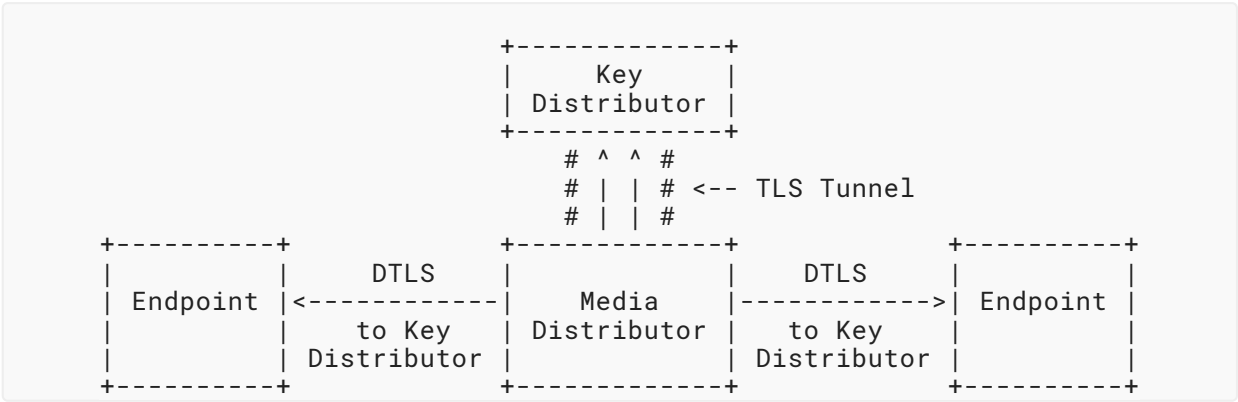

Figure 1: TLS Tunnel to Key Distributor

The three entities involved in this communication flow are the endpoint, the Media Distributor, and the Key Distributor. The behavior of each entity is described in Section 5.

The Key Distributor is a logical function that might be co-resident with a key management server operated by an enterprise, might reside in one of the endpoints participating in the conference, or might reside at some other location that is trusted with E2E keying material.

4. Example Message Flows

This section provides an example message flow to help clarify the procedures described later in this document. It is necessary that the Key Distributor and Media Distributor establish a mutually authenticated TLS connection for the purpose of sending tunneled messages, though the complete TLS handshake for the tunnel is not shown in Figure 2 because there is nothing new this document introduces with regard to those procedures.

Once the tunnel is established, it is possible for the Media Distributor to relay the DTLS messages between the endpoint and the Key Distributor. Figure 2 shows a message flow wherein the endpoint uses DTLS-SRTP to establish an association with the Key Distributor. In the process, the Media Distributor shares its supported SRTP protection profile information (see [RFC5764]), and the Key Distributor shares the HBH keying material and selected cipher with the Media Distributor.

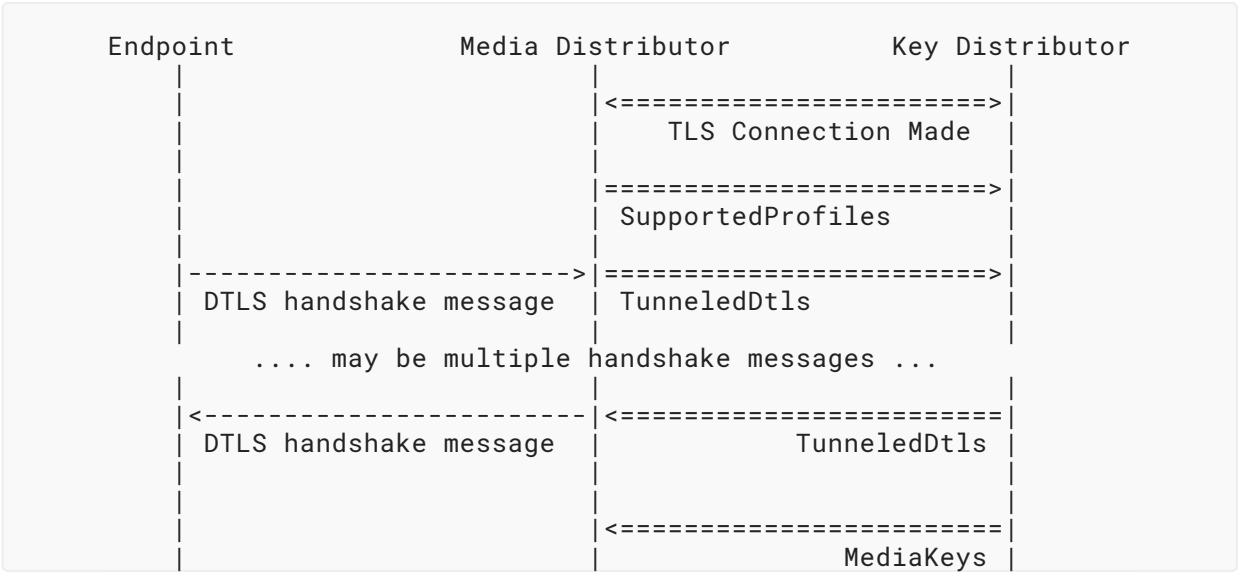

Figure 2: Sample DTLS-SRTP Exchange via the Tunnel

After the initial TLS connection has been established, each of the messages on the right-hand side of Figure 2 is a tunneling protocol message, as defined in Section 6.

SRTP protection profiles supported by the Media Distributor will be sent in a `SupportedProfiles` message when the TLS tunnel is initially established. The Key Distributor will use that information to select a common profile supported by both the endpoint and the Media Distributor to ensure that HBH operations can be successfully performed.

As DTLS messages are received from the endpoint by the Media Distributor, they are forwarded to the Key Distributor encapsulated inside a `TunneledDtls` message. Likewise, as `TunneledDtls` messages are received by the Media Distributor from the Key Distributor, the encapsulated DTLS packet is forwarded to the endpoint.

The Key Distributor will provide the SRTP keying material [RFC3711] to the Media Distributor for HBH operations via the `MediaKeys` message. The Media Distributor will extract this keying material from the `MediaKeys` message when received and use it for HBH encryption and authentication.

5. Tunneling Procedures

The following subsections explain in detail the expected behavior of the endpoint, the Media Distributor, and the Key Distributor.

It is important to note that the tunneling protocol described in this document is not an extension to TLS or DTLS. Rather, it is a protocol that transports DTLS messages generated by an endpoint or Key Distributor as data inside of the TLS connection established between the Media Distributor and Key Distributor.

5.1. Endpoint Procedures

The endpoint follows the procedures outlined for DTLS-SRTP [RFC5764] in order to establish the cipher and keys used for encryption and authentication, with the endpoint acting as the client and the Key Distributor acting as the server. The endpoint does not need to be aware of the fact that DTLS messages it transmits toward the Media Distributor are being tunneled to the Key Distributor.

The endpoint **MUST** include a unique identifier in the `tls-id` Session Description Protocol (SDP) attribute [RFC8866] in all offer and answer messages [RFC3264] that it generates, as per [RFC8842]. Further, the endpoint **MUST** include this same unique identifier in the `external_session_id` extension [RFC8844] in the `ClientHello` message when establishing a DTLS association.

When receiving an `external_session_id` value from the Key Distributor, the client **MUST** check to ensure that value matches the `tls-id` value received in SDP. If the values do not match, the endpoint **MUST** consider any received keying material to be invalid and terminate the DTLS association.

5.2. Tunnel Establishment Procedures

Either the Media Distributor or Key Distributor initiates the establishment of a TLS tunnel. Which entity acts as the TLS client when establishing the tunnel and what event triggers the establishment of the tunnel are outside the scope of this document. Further, how the trust relationships are established between the Key Distributor and Media Distributor are also outside the scope of this document.

A tunnel **MUST** be a mutually authenticated TLS connection.

The Media Distributor or Key Distributor **MUST** establish a tunnel prior to forwarding tunneled DTLS messages. Given the time-sensitive nature of DTLS-SRTP procedures, a tunnel **SHOULD** be established prior to the Media Distributor receiving a DTLS message from an endpoint.

A single tunnel **MAY** be used to relay DTLS messages between any number of endpoints and the Key Distributor.

A Media Distributor **MAY** have more than one tunnel established between itself and one or more Key Distributors. When multiple tunnels are established, which tunnel or tunnels to use to send messages for a given conference is outside the scope of this document.

5.3. Media Distributor Tunneling Procedures

The first message transmitted over the tunnel is the `SupportedProfiles` message (see Section 6). This message informs the Key Distributor about which DTLS-SRTP profiles the Media Distributor supports. This message **MUST** be sent each time a new tunnel connection is established or, in the case of connection loss, when a connection is re-established. The Media Distributor **MUST** support the same list of protection profiles for the duration of any endpoint-initiated DTLS association and tunnel connection.

The Media Distributor **MUST** assign a unique association identifier for each endpoint-initiated DTLS association and include it in all messages forwarded to the Key Distributor. The Key Distributor will subsequently include this identifier in all messages it sends so that the Media Distributor can map messages received via a tunnel and forward those messages to the correct endpoint. The association identifier **MUST** be a version 4 Universally Unique Identifier (UUID), as described in Section 4.4 of [RFC4122].

When a DTLS message is received by the Media Distributor from an endpoint, it forwards the UDP payload portion of that message to the Key Distributor encapsulated in a `TunneledDtls` message. The Media Distributor is not required to forward all messages received from an endpoint for a given DTLS association through the same tunnel if more than one tunnel has been established between it and a Key Distributor.

When a `MediaKeys` message is received, the Media Distributor **MUST** extract the cipher and keying material conveyed in order to subsequently perform HBH encryption and authentication operations for RTP and RTP Control Protocol (RTCP) packets sent between it and an endpoint. Since the HBH keying material will be different for each endpoint, the Media Distributor uses the association identifier included by the Key Distributor to ensure that the HBH keying material is used with the correct endpoint.

The Media Distributor **MUST** forward all DTLS messages received from either the endpoint or the Key Distributor (via the `TunneledDtls` message) to ensure proper communication between those two entities.

When the Media Distributor detects an endpoint has disconnected or when it receives conference control messages indicating the endpoint is to be disconnected, the Media Distributor **MUST** send an `EndpointDisconnect` message with the association identifier assigned to the endpoint to the Key Distributor. The Media Distributor **SHOULD** take a loss of all RTP and RTCP packets as an indicator that the endpoint has disconnected. The particulars of how RTP and RTCP are to be used to detect an endpoint disconnect, such as timeout period, are not specified. The Media Distributor **MAY** use additional indicators to determine when an endpoint has disconnected.

5.4. Key Distributor Tunneling Procedures

Each TLS tunnel established between the Media Distributor and the Key Distributor **MUST** be mutually authenticated.

When the Media Distributor relays a DTLS message from an endpoint, the Media Distributor will include an association identifier that is unique per endpoint-originated DTLS association. The association identifier remains constant for the life of the DTLS association. The Key Distributor identifies each distinct endpoint-originated DTLS association by the association identifier.

When processing an incoming endpoint association, the Key Distributor **MUST** extract the `external_session_id` value transmitted in the `ClientHello` message and match that against the `tls-id` value the endpoint transmitted via SDP. If the values in SDP and the `ClientHello` message do not match, the DTLS association **MUST** be rejected.

The process through which the `tls-id` value in SDP is conveyed to the Key Distributor is outside the scope of this document.

The Key Distributor **MUST** match the fingerprint of the certificate and `external_session_id` [RFC8844] received from the endpoint via DTLS with the expected fingerprint [RFC8122] and `tls-id` [RFC8842] values received via SDP. It is through this process that the Key Distributor can be sure to deliver the correct conference key to the endpoint.

The Key Distributor **MUST** report its own unique identifier in the `external_session_id` extension. This extension is sent in the `EncryptedExtensions` message in DTLS 1.3 and the `ServerHello` message in previous DTLS versions. This value **MUST** also be conveyed back to the client via SDP as a `tls-id` attribute.

The Key Distributor **MUST** encapsulate any DTLS message it sends to an endpoint inside a `TunneledDtls` message (see Section 6). The Key Distributor is not required to transmit all messages for a given DTLS association through the same tunnel if more than one tunnel has been established between it and the Media Distributor.

The Key Distributor **MUST** use the same association identifier in messages sent to an endpoint as was received in messages from that endpoint. This ensures the Media Distributor can forward the messages to the correct endpoint.

The Key Distributor extracts tunneled DTLS messages from an endpoint and acts on those messages as if that endpoint had established the DTLS association directly with the Key Distributor. The Key Distributor is acting as the DTLS server, and the endpoint is acting as the DTLS client. The handling of the messages and certificates is exactly the same as normal DTLS-SRTP procedures between endpoints.

The Key Distributor **MUST** send a `MediaKeys` message to the Media Distributor immediately after the DTLS handshake completes. The `MediaKeys` message includes the selected cipher (i.e., protection profile), Master Key Identifier (MKI) value [RFC3711] (if any), HBH SRTP master keys, and SRTP master salt values. The Key Distributor **MUST** use the same association identifier in the `MediaKeys` message as is used in the `TunneledDtls` messages for the given endpoint.

There are presently two SRTP protection profiles defined for PERC, namely `DOUBLE_AEAD_AES_128_GCM_AEAD_AES_128_GCM` and `DOUBLE_AEAD_AES_256_GCM_AEAD_AES_256_GCM` [RFC8723]. As explained in Section 5.2 of [RFC8723], the Media Distributor is only given the SRTP master key for HBH operations. As such, the SRTP master key length advertised in the `MediaKeys` message is half the length of the key normally associated with the selected "double" protection profile.

The Key Distributor uses the certificate fingerprint of the endpoint along with the unique identifier received in the `external_session_id` extension to determine with which conference a given DTLS association is associated.

The Key Distributor **MUST** select a cipher that is supported by itself, the endpoint, and the Media Distributor to ensure proper HBH operations.

When the DTLS association between the endpoint and the Key Distributor is terminated, regardless of which entity initiated the termination, the Key Distributor **MUST** send an `EndpointDisconnect` message with the association identifier assigned to the endpoint to the Media Distributor.

5.5. Versioning Considerations

Since the Media Distributor sends the first message over the tunnel, it effectively establishes the version of the protocol to be used. If that version is not supported by the Key Distributor, the Key Distributor **MUST** transmit an `UnsupportedVersion` message containing the highest version number supported and close the TLS connection.

The Media Distributor **MUST** take note of the version received in an `UnsupportedVersion` message and use that version when attempting to re-establish a failed tunnel connection. Note that it is not necessary for the Media Distributor to understand the newer version of the protocol to understand that the first message received is an `UnsupportedVersion` message. The Media Distributor can determine from the first four octets received what the version number is and that the message is an `UnsupportedVersion` message. The rest of the data received, if any, would be discarded and the connection closed (if not already closed).

6. Tunneling Protocol

Tunneled messages are transported via the TLS tunnel as application data between the Media Distributor and the Key Distributor. Tunnel messages are specified using the format described in [RFC8446], Section 3. As in [RFC8446], all values are stored in network byte (big endian) order; the uint32 represented by the hex bytes 01 02 03 04 is equivalent to the decimal value 16909060.

This protocol defines several different messages, each of which contains the following information:

- message type identifier
- message body length
- the message body

Each of the tunnel messages is a `TunnelMessage` structure with the message type indicating the actual content of the message body.

6.1. TunnelMessage Structure

`TunnelMessage` defines the structure of all messages sent via the tunnel protocol. That structure includes a field called `msg_type` that identifies the specific type of message contained within `TunnelMessage`.

```
enum {
    supported_profiles(1),
    unsupported_version(2),
    media_keys(3),
    tunneled_dtls(4),
    endpoint_disconnect(5),
    (255)
} MsgType;

opaque uuid[16];

struct {
    MsgType msg_type;
    uint16 length;
    select (MsgType) {
        case supported_profiles:  SupportedProfiles;
        case unsupported_version: UnsupportedVersion;
        case media_keys:          MediaKeys;
        case tunneled_dtls:       TunneledDtls;
        case endpoint_disconnect: EndpointDisconnect;
  } body;
} TunnelMessage;
```

The elements of `TunnelMessage` include:

`msg_type`: the type of message contained within the structure `body`.

`length`: the length in octets of the following `body` of the message.

`body`: the actual message being conveyed within this `TunnelMessage` structure.

6.2. SupportedProfiles Message

The `SupportedProfiles` message is defined as:

```
uint8 SRTPProtectionProfile[2]; /* from RFC 5764 */

struct {
    uint8 version;
    SRTPProtectionProfile protection_profiles<2..2^16-1>;
} SupportedProfiles;
```

The elements of `SupportedProfiles` include:

`version`: this document specifies version 0x00.

`protection_profiles`: the list of two-octet SRTP protection profile values, as per [RFC5764], supported by the Media Distributor.

6.3. UnsupportedVersion Message

The `UnsupportedVersion` message is defined as:

```
struct {
    uint8 highest_version;
} UnsupportedVersion;
```

`UnsupportedVersion` contains this single element:

`highest_version`: indicates the highest version of the protocol supported by the Key Distributor.

6.4. MediaKeys Message

The `MediaKeys` message is defined as:

```
struct {
    uuid association_id;
    SRTPProtectionProfile protection_profile;
    opaque mki<0..255>;
    opaque client_write_SRTP_master_key<1..255>;
    opaque server_write_SRTP_master_key<1..255>;
    opaque client_write_SRTP_master_salt<1..255>;
    opaque server_write_SRTP_master_salt<1..255>;
} MediaKeys;
```

The fields are described as follows:

`association_id`: a value that identifies a distinct DTLS association between an endpoint and the Key Distributor.

`protection_profiles`: the value of the two-octet SRTP protection profile value, as per [RFC5764], used for this DTLS association.

`mki`: master key identifier [RFC3711]; a zero-length field indicates that no MKI value is present.

`client_write_SRTP_master_key`: the value of the SRTP master key used by the client (endpoint).

`server_write_SRTP_master_key`: the value of the SRTP master key used by the server (Media Distributor).

`client_write_SRTP_master_salt`: the value of the SRTP master salt used by the client (endpoint).

`server_write_SRTP_master_salt`: the value of the SRTP master salt used by the server (Media Distributor).

6.5. TunneledDtls Message

The `TunneledDtls` message is defined as:

```
struct {
    uuid association_id;
    opaque dtls_message<1..2^16-1>;
} TunneledDtls;
```

The fields are described as follows:

`association_id`: a value that identifies a distinct DTLS association between an endpoint and the Key Distributor.

`dtls_message`: the content of the DTLS message received by the endpoint or to be sent to the endpoint, including one or more complete DTLS records.

6.6. EndpointDisconnect Message

The `EndpointDisconnect` message is defined as:

```
struct {
    uuid association_id;
} EndpointDisconnect;
```

The field is described as follows:

`association_id`: a value that identifies a distinct DTLS association between an endpoint and the Key Distributor.

7. Example Binary Encoding

The `TunnelMessage` is encoded in binary, following the procedures specified in [RFC8446]. This section provides an example of what the bits on the wire would look like for the `SupportedProfiles` message that advertises support for both `DOUBLE_AEAD_AES_128_GCM_AEAD_AES_128_GCM` and `DOUBLE_AEAD_AES_256_GCM_AEAD_AES_256_GCM` [RFC8723].

```
TunnelMessage:
         message_type: 0x01
               length: 0x0007
    SupportedProfiles:
                  version:  0x00
      protection_profiles:  0x0004 (length)
                            0x0009000A (value)
```

Thus, the encoding on the wire, presented here in network byte order, would be this stream of octets:

```
0x0100070000040009000A
```

8. IANA Considerations

This document establishes the "Datagram Transport Layer Security (DTLS) Tunnel Protocol Message Types for Privacy Enhanced Conferencing" registry to contain message type values used in the DTLS tunnel protocol. These message type values are a single octet in length. This document defines the values shown in Table 1 below, leaving the balance of possible values reserved for future specifications:

MsgType	Description
0x01	Supported SRTP Protection Profiles
0x02	Unsupported Version
0x03	Media Keys
0x04	Tunneled DTLS
0x05	Endpoint Disconnect

Table 1: Message Type Values for the DTLS Tunnel Protocol

The value 0x00 is reserved, and all values in the range 0x06 to 0xFF are available for allocation. The procedures for updating this table are those defined as "IETF Review" in Section 4.8 of [RFC8126].

9. Security Considerations

Since the procedures in this document rely on TLS [RFC8446] for transport security, the security considerations for TLS should be reviewed when implementing the protocol defined in this document.

While the tunneling protocol defined in this document does not use DTLS-SRTP [RFC5764] directly, it does convey and negotiate some of the same information (e.g., protection profile data). As such, a review of the security considerations found in that document may be useful.

This document describes a means of securely exchanging keying material and cryptographic transforms for both E2E and HBH encryption and authentication of media between an endpoint and a Key Distributor via a Media Distributor. Additionally, the procedures result in delivering HBH information to the intermediary Media Distributor. The Key Distributor and endpoint are the only two entities with access to both the E2E and HBH keys, while the Media Distributor has access to only HBH information. Section 8.2 of [RFC8871] enumerates various attacks against which one must guard when implementing a Media Distributor; these scenarios are important to note.

A requirement in this document is that a TLS connection between the Media Distributor and the Key Distributor be mutually authenticated. The reason for this requirement is to ensure that only an authorized Media Distributor receives the HBH keying material. If an unauthorized Media Distributor gains access to the HBH keying material, it can easily cause service degradation or denial by transmitting HBH-valid packets that ultimately fail E2E authentication or replay protection checks (see Section 3.3.2 of [RFC3711]). Even if service does not appear degraded in any way, transmitting and processing bogus packets are a waste of both computational and network resources.

The procedures defined in this document assume that the Media Distributor will properly convey DTLS messages between the endpoint and Key Distributor. Should it fail in that responsibility by forwarding DTLS messages from endpoint A advertised as being from endpoint B, this will result in a failure at the DTLS layer of those DTLS sessions. This could be an additional attack vector that Key Distributor implementations should consider.

While E2E keying material passes through the Media Distributor via the protocol defined in this document, the Media Distributor has no means of gaining access to that information and therefore cannot affect the E2E media processing function in the endpoint except to present it with invalid or replayed data. That said, any entity along the path that interferes with the DTLS exchange between the endpoint and the Key Distributor, including a malicious Media Distributor that is not properly authorized, could prevent an endpoint from properly communicating with the Key Distributor and therefore prevent successful conference participation.

It is worth noting that a compromised Media Distributor can convey information to an adversary, such as participant IP addresses, negotiated protection profiles, or other metadata. While [RFC8871] explains that a malicious or compromised Media Distributor can disrupt communications, an additional attack vector introduced by this protocol is the potential disruption of DTLS negotiation or premature removal of a participant from a conference by sending an `EndpointDisconnect` message to the Key Distributor.

The Key Distributor should be aware of the possibility that a malicious Media Distributor might transmit an `EndpointDisconnect` message to the Key Distributor when the endpoint is in fact still connected.

While the Security Considerations section of [RFC8871] describes various attacks one needs to consider with respect to the Key Distributor and denial of service, use of this protocol introduces another possible attack vector. Consider the case where a malicious endpoint sends unsolicited DTLS-SRTP messages to a Media Distributor. The Media Distributor will normally forward those messages to the Key Distributor and, if found invalid, such messages only serve to consume resources on both the Media Distributor and Key Distributor.

10. References

10.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC8122] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <https://www.rfc-editor.org/info/rfc8122>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8723] Jennings, C., Jones, P., Barnes, R., and A.B. Roach, "Double Encryption Procedures for the Secure Real-Time Transport Protocol (SRTP)", RFC 8723, DOI 10.17487/RFC8723, April 2020, <https://www.rfc-editor.org/info/rfc8723>.

[RFC8842] Holmberg, C. and R. Shpount, "Session Description Protocol (SDP) Offer/Answer Considerations for Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS)", RFC 8842, DOI 10.17487/RFC8842, January 2021, <https://www.rfc-editor.org/info/rfc8842>.

[RFC8844] Thomson, M. and E. Rescorla, "Unknown Key-Share Attacks on Uses of TLS with the Session Description Protocol (SDP)", RFC 8844, DOI 10.17487/RFC8844, January 2021, <https://www.rfc-editor.org/info/rfc8844>.

[RFC8871] Jones, P., Benham, D., and C. Groves, "A Solution Framework for Private Media in Privacy-Enhanced RTP Conferencing (PERC)", RFC 8871, DOI 10.17487/RFC8871, January 2021, <https://www.rfc-editor.org/info/rfc8871>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

10.2. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

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

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/info/rfc8866>.

Acknowledgements

The authors would like to thank David Benham and Cullen Jennings for reviewing this document and providing constructive comments.

Authors' Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, North Carolina 27709
United States of America
Phone: +1 919 476 2048
Email: paulej@packetizer.com

Paul M. Ellenbogen
Princeton University
Phone: +1 206 851 2069
Email: pe5@cs.princeton.edu

Nils H. Ohlmeier
8x8, Inc.
Phone: +1 408 659 6457
Email: nils@ohlmeier.org