DTLS Tunnel for PERC

draft-jones-perc-dtls-tunnel-04

Presenter: Adam Roach, Mozilla

Paul E. Jones, Cisco
Paul M. Ellenbogen, Princeton
Nils H. Ohlmeier, Mozilla

IETF 97 • November 2016
Notable Changes

• Switched tunnel transport from DTLS to TLS
• Changed key field names to align with RFC 5764 (DTLS-SRTP)
• Introduced a conference identifier field
• Switched back to a TLS-style syntax (similar to draft -00)
 TLS-Style Syntax

Message primitives are

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

Common message structure

```c
struct {
    uint8 version;
    MsgType msg_type;
    select (MsgType) {
        case unsupported_version: UnsupportedVersion;
        case supported_profiles: SupportedProfiles;
        case media_keys: MediaKeys;
        case tunneled_dtls: TunneledDtls;
        case endpoint_disconnect: EndpointDisconnect;
    } body;
} TunnelMessage;
```
High-Level Message Sequence

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>media distributor</th>
<th>key distributor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TLS Connection Made</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SupportedProfiles</td>
</tr>
<tr>
<td></td>
<td>DTLS handshake message</td>
<td>TunneledDtls</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MediaKeys</td>
</tr>
<tr>
<td></td>
<td>.... may be multiple</td>
<td></td>
</tr>
<tr>
<td></td>
<td>handshake messages</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;=------------------------</td>
<td>&lt;=------------------------</td>
</tr>
<tr>
<td></td>
<td>DTLS handshake message</td>
<td>TunneledDtls</td>
</tr>
</tbody>
</table>
SupportedProfiles

• Message sent from the Media Distributor to the Key Distributor to indicate which hop-by-hop SRTP encryption & authentication algorithms are supported

```c
uint8 SRTPProtectionProfile[2]; /* from RFC5764 */

struct {
    SRTPProtectionProfile protection_profiles<0..2^16-1>;
} SupportedProfiles;
```
TunneledDtls

• This message is used to tunnel DTLS packets between the media distributor and the key distributor

```c
struct {
    uint32 association_id;
    opaque conf_id<0..255>;
    opaque dtls_message<0..2^16-1>;
} TunneledDtls;
```

• The conference ID allows the transmitter to indicate the conference to which a tunneled message belongs (more later)
MediaKeys

• Allows the key distributor to provide the media distributor with hop-by-hop keying material and selected cipher

```c
struct {
    uint32 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>;
    opaque conf_id<0..255>;
} MediaKeys;
```

• Note the conference identifier is present here, too (more later)
EndpointDisconnect

• This message is sent from the media distributor to the key distributor to provide a clear indication that the associated endpoint is no longer a conference participant

```c
struct {
    uint32 association_id;
} EndpointDisconnect;
```
UnsupportedVersion

- Sent by the key distributor to indicate to the media distributor that the version of the protocol advertised is not supported
- Media distributor is responsible for moving to the version supported by the key distributor

```c
struct { } UnsupportedVersion;
```
Alice attempts to attend two different, overlapping meetings, initiating a DTLS associations for each of those.

Which DTLS association belongs to conference “A” and conference “B”? This determines which “EKT Key” to return.
Conference Identification

• Assumption: the key distributor knows which users (including the user’s certificate fingerprint) are allowed to be given a given conference key

• If an endpoint uses the same certificate, we have a problem to solve
  • Solution: put the conference identifier into the TunneledDtls message sent by the media distributor, allowing the key distributor to be able to associate a DTLS association with a particular conference
  • Preference: the media distributor not have to know a conference identifier a priori and require that each simultaneous call use a different certificate, thus allowing the key distributor to determine which key to use based on the certificate fingerprint
  • Alternative: advertise a conference identifier in the DTLS handshake (discovery external to PERC)

• How does the media distributor know how to put a user’s media flows into a given conference?
  • Solution: put a conference identifier into the MediaKeys message, effectively allowing the key distributor to tell the media distributor how to group user flows into a conference
  • Preference: not have this field and accept that some higher-level call control function instructs the media distributor on how to associate flows into a conference (outside the scope of this protocol or PERC entirely)