        A Solution Framework for Private Media in Privacy Enhanced RTP
                              Conferencing
                  draft-ietf-perc-private-media-framework-01

Abstract

   This document describes a solution framework for ensuring that media
   confidentiality and integrity are maintained end-to-end within the
   context of a switched conferencing environment where media
   distribution devices are not trusted with the end-to-end media
   encryption keys. The solution aims to build upon existing security
   mechanisms defined for the real-time transport protocol (RTP).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Switched conferencing is an increasingly popular model for multimedia
   conferences with multiple participants using a combination of audio,
   video, text, and other media types. With this model, real-time media

flows from conference participants are not mixed, transcoded,
transrated, recomposed, or otherwise manipulated by a Media
Distributor, as might be the case with a traditional media server or
multipoint control unit (MCU). Instead, media flows transmitted by
conference participants are simply forwarded by the Media Distributor
to each of the other participants, often forwarding only a subset of
flows based on voice activity detection or other criteria. In some
instances, the Media Distributors may make limited modifications to
RTP [RFC3550] headers, for example, but the actual media content
(e.g., voice or video data) is unaltered.

An advantage of switched conferencing is that Media Distributors can
be more easily deployed on general-purpose computing hardware,
including virtualized environments in private and public clouds.
Deploying conference resources in a public cloud environment might
introduce a higher security risk. Whereas traditional conference
resources were usually deployed in private networks that were
protected, cloud-based conference resources might be viewed as less
secure since they are not always physically controlled by those who
use them. Additionally, there are usually several ports open to the
public in cloud deployments, such as for remote administration, and
so on.

This document defines a solution framework wherein media privacy is
ensured by making it impossible for a media distribution device to
gain access to keys needed to decrypt or authenticate the actual
media content sent between conference participants. At the same
time, the framework allows for the Media Distributors to modify
certain RTP headers; add, remove, encrypt, or decrypt RTP header
extensions; and encrypt and decrypt RTCP packets. The framework also
prevents replay attacks by authenticating each packet transmitted
between a given participant and the media distribution device using a
unique key per endpoint that is independent from the key for media
encryption and authentication.

A goal of this document is to define a framework for enhanced privacy
in RTP-based conferencing environments while utilizing existing
security procedures defined for RTP with minimal enhancements.

2. Conventions Used in This Document

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] when they
appear in ALL CAPS. These words may also appear in this document in
lower case as plain English words, absent their normative meanings.

Additionally, this solution framework uses the following conventions, terms and acronyms:

End-to-End (E2E): Communications from one endpoint through one or more Media Distribution Devices to the endpoint at the other end.

Hop-by-Hop (HBH): Communications between an endpoint and a Media Distribution Device or between Media Distribution Devices.

Endpoint: An RTP flow terminating entity that has possession of E2E media encryption keys and terminates E2E encryption. This may include embedded user conferencing equipment or browsers on computers, media gateways, MCUs, media recording device and more that are in the trusted domain for a given deployment.

Media Distributor (MD): An RTP middlebox that is not allowed to to have access to E2E encryption keys. It may operate according to any of the RTP topologies [I-D.ietf-avtcore-rtp-topologies-update] per the constraints defined by the PERC system, which includes, but not limited to, having no access to RTP media unencrypted and having limits on what RTP header field it can alter.

Key Distributor: An entity that is a logical function which passes keying material and related information to endpoints and Media Distributor(s) that is appropriate for each. The Key Distributor might be co-resident with another entity trusted with E2E keying material.

Conference: Two or more participants communicating via trusted endpoints to exchange RTP flows through one or more Media Distributor.

Third Party: Any entity that is not an Endpoint, Media Distributor, Key Distributor or Call Processing entity as described in this document.

3. PERC Entities and Trust Model

The following figure depicts the trust relationships, direct or indirect, between entities described in the subsequent sub-sections. Note that these entities may be co-located or further divided into multiple, separate physical devices.

Please note that some entities classified as untrusted in the simple, general deployment scenario used most commonly in this document might be considered trusted in other deployments. This document does not preclude such scenarios, but will keep the definitions and examples

focused by only using the the simple, most general deployment
scenario.

```
                              |
        +----------+          |          +-----------------+
        | Endpoint |          |          | Call Processing |
        +----------+          |          +-----------------+
                              |
                              |
        +----------------+    |          +-------------------+
        | Key Distributor|    |          | Media Distributor |
        +----------------+    |          +-------------------+
                              |
             Trusted          |               Untrusted
             Entities         |               Entities
                              |
```
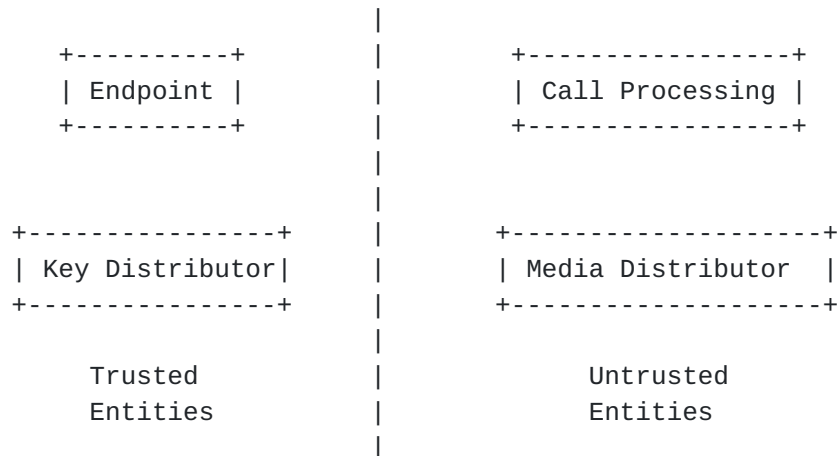
Figure 1: Trusted and Untrusted Entities in PERC

3.1. Untrusted Entities

The architecture described in this framework document enables
conferencing infrastructure to be hosted in domains, such as in a
cloud conferencing provider's facilities, where the trustworthiness
is below the level needed to assume the privacy of participant's
media will not be compromised. The conferencing infrastructure in
such a domain is still trusted with reliably connecting the
participants together in a conference, but not trusted with keying
material needed to decrypt any of the participant's media. Entities
in such lower trustworthiness domains will simply be referred to as
untrusted entities from this point forward. This does not mean that
they are completely untrusted as they may be trusted with most non-
media related aspects of hosting a conference.

3.1.1. Media Distributor

A Media Distributor forwards RTP flows between endpoints in the
conference while performing per-hop authentication of each RTP
packet. The Media Distributor may need access to one or more RTP
headers or header extensions, potentially adding or modifying a
certain subset. The Media Distributor will also relay secured
messaging between the endpoints and the Key Distributor and will
acquire per-hop key information from the Key Distributor. The actual
media content MUST NOT not be decryptable by a Media Distributor, so
it is untrusted to have access to the E2E media encryption keys,
which this framework's key exchange mechanisms will prevent.

An endpoint's ability to join a conference hosted by a Media
Distributor MUST NOT alone be interpreted as being authorized to have
access to the E2E media encryption keys, as the Media Distributor
does not have the ability to determine whether an endpoint is
authorized.

A Media Distributor MUST perform its role in properly forwarding
media packets while taking measures to mitigate the adverse effects
of denial of service attacks (refer to Section 6), etc, to a level
equal to or better than traditional conferencing (i.e. pre-PERC)
deployments.

A Media Distributor or associated conferencing infrastructure may
also initiate or terminate various conference control related
messaging, which is outside the scope of this framework document.

3.1.2. Call Processing

The call processing function is untrusted in the simple, general
deployment scenario. When a physical subset of the call processing
function resides in facilities outside the trusted domain, it should
not be trusted to have access to E2E key information.

The call processing function may include the processing of call
signaling messages, as well as the signing of those messages. It may
also authenticate the endpoints for the purpose of call signaling and
subsequently joining of a conference hosted through one or more Media
Distributors. Call processing may optionally ensure the privacy of
call signaling messages between itself, the endpoint, and other
entities.

In any deployment scenario where the call processing function is
considered trusted, the call processing function MUST ensure the
integrity of received messages before forwarding to other entities.

3.2. Trusted Entities

From the PERC model system perspective, entities considered trusted
(refer to Figure 1) can be in possession of the E2E media encryption
key(s) for one or more conferences.

3.2.1. Endpoint

An endpoint is considered trusted and will have access to E2E key
information. While it is possible for an endpoint to be compromised,
subsequently performing in undesired ways, defining endpoint
resistance to compromise is outside the scope of this document.
Endpoints will take measures to mitigate the adverse effects of

denial of service attacks (refer to Section 6) from other entities,
including from other endpoints, to a level equal to or better than
traditional conference (i.e., pre-PERC) deployments.

3.2.2. Key Distributor

The Key Distributor, which may be collocated with an endpoint or
exist standalone, is responsible for providing key information to
endpoints for both end-to-end and hop-by-hop security and for
providing key information to Media Distributors for the hop-by-hop
security.

Interaction between the Key Distributor and the call processing
function may be necessary to for proper conference-to-endpoint
mappings, which may or may not be satisfied by getting information
directly from the endpoints or via some other means. See Section 7
for a related item in the To Do list.

The Key Distributor needs to be secured and managed in a way to
prevent exploitation by an adversary, as any kind of compromise of
the Key Distributor puts the security of the conference at risk.

4. Framework for PERC

The purpose for this framework is to define a means through which
media privacy can be ensured when communicating within a conferencing
environment consisting of one or more Media Distributors that only
switch, hence not terminate, media. It does not otherwise attempt to
hide the fact that a conference between endpoints is taking place.

This framework reuses several specified RTP security technologies,
including SRTP [RFC3711], PERC EKT [I-D.ietf-perc-srtp-ekt-diet], and
DTLS-SRTP [RFC5764].

4.1. End-to-End and Hop-by-Hop Authenticated Encryption

This solution framework focuses on the end-to-end privacy and
integrity of the participant's media by limiting access to end-to-end
key information to trusted entities. However, this framework does
give a Media Distributor access to RTP headers and all or most header
extensions, as well as the ability to modify a certain subset of
those headers and to add header extensions. Packets received by a
Media Distributor or an endpoint are authenticated hop-by-hop.

To enable all of the above, this framework defines the use of two
security contexts and two associated encryption keys; an "inner" key
(E2E Key(i); i={a given endpoint}) for authenticated encryption of
RTP media between endpoints and an "outer" key (HBH Key(j); j={a

given hop}) for the hop between an endpoint and a Media Distributor
or between Media Distributor. Reference the following figure.

```
    +-------------+                                  +-------------+
    |             |################################|             |
    |    Media    |------------------------------->|    Media    |
    | Distributor |<-------------------------------| Distributor |
    |      X      |################################|      Y      |
    |             |           HBH Key (XY)          |             |
    +-------------+                                  +-------------+
      #  ^ |  #                                        #  ^ |  #
      #  | |  #        HBH              HBH            #  | |  #
      #  | |  # <== Key(AX)         Key(YB) ==> #  | |  #
      #  | |  #                                        #  | |  #
      #  |<+--#---- E2E Key (A)      E2E Key (B) ---#->|  |  #
      #  | |  #                                        #  | |  #
      #  | v  #                                        #  | v  #
    +-------------+                                  +-------------+
    | Endpoint A  |                                  | Endpoint B  |
    +-------------+                                  +-------------+
```
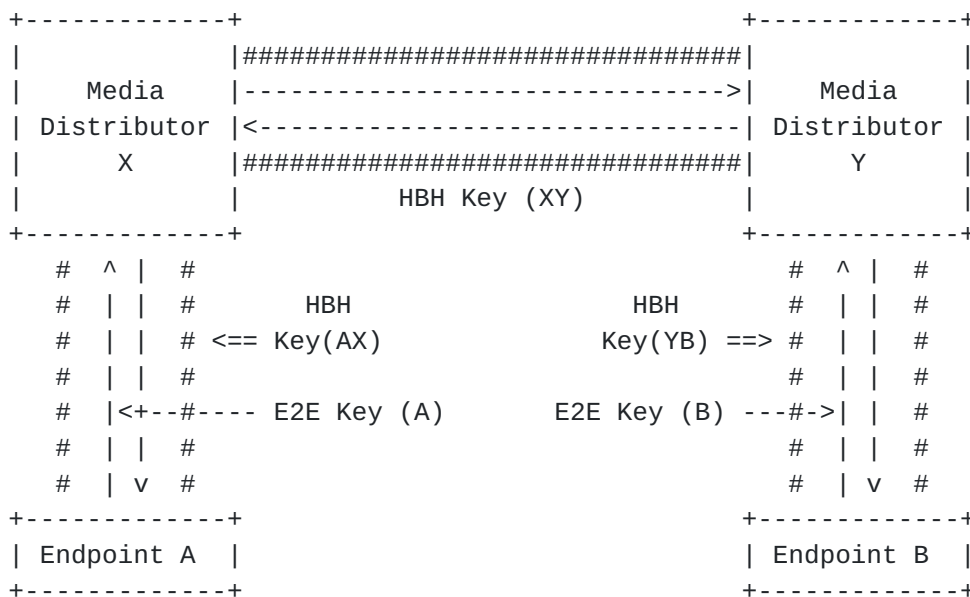
          E2E and HBH Keys Used for Authenticated Encryption

   The PERC Double draft specification [I-D.ietf-perc-double] uses
   standard SRTP keying material and recommended cryptographic
   transform(s) to first form the inner, end-to-end SRTP cryptographic
   context. That end-to-end SRTP cryptographic context MAY be used to
   encrypt some RTP header extensions along with RTP media content. The
   output of this is treated like an RTP packet and encrypted again
   using the outer hop-by-hop cryptographic context. The endpoint
   executes the entire Double operation while the Media Distributor just
   performs the outer, hop-by-hop operation.

   RTCP can only be encrypted hop-by-hop, not end-to-end. This
   framework introduces no additional step for RTCP authenticated
   encryption, so the procedures needed are specified in [RFC3711] and
   use the same outer, hop-by-hop cryptographic context chosen in the
   Double operation described above.

4.2. E2E Key Confidentiality

   To ensure the confidentiality of E2E keys shared between endpoints,
   endpoints will make use of a common Key Encryption Key (KEK) that is
   known only by the trusted entities in a conference. That KEK,
   defined in the PERC EKT [I-D.ietf-perc-srtp-ekt-diet] as the EKT Key,
   will be used to subsequently encrypt SRTP master keys used for E2E
   authenticated encryption (E2E Key(i); i={a given endpoint}) of media
   sent by a given endpoint.

```
+---------------------+-----------+-------+-------+------------+
| Key      /    Entity | Endpoint A |  MD X |  MD Y | Endpoint B |
+---------------------+-----------+-------+-------+------------+
| KEK                 |    Yes    |  No   |  No   |    Yes     |
+---------------------+-----------+-------+-------+------------+
| E2E Key (i)         |    Yes    |  No   |  No   |    Yes     |
+---------------------+-----------+-------+-------+------------+
| HBH Key (A<=>MD X)  |    Yes    |  Yes  |  No   |    No      |
+---------------------+-----------+-------+-------+------------+
| HBH Key (B<=>MD Y)  |    No     |  No   |  Yes  |    Yes     |
+---------------------+-----------+-------+-------+------------+
```

Figure 2: Keys per Entity

4.3. E2E Keys and Endpoint Operations

Any given RTP media flow can be identified by its SSRC, and endpoints
might send more than one at a time and change the mix of media flows
transmitted during the life of a conference.

Thus, endpoints MUST maintain a list of SSRCs from received RTP flows
and each SSRC's associated E2E Key(i) information. Following a
change of the KEK (i.e., EKT Key), prior E2E Key(i) information
SHOULD be retained for a period long enough to ensure that late-
arriving or out-of-order packets from other endpoints can be
successfully decrypted. The endpoint MUST discard the E2E Key(i) and
KEK information no later than when it leaves the conference.

If there is a need to encrypt one or more RTP header extensions end-
to-end, an encryption key is derived from the end-to-end SRTP master
key to encrypt header extensions as per [RFC6904]. The Media
Distributor will not be able use the information contained in those
header extensions encrypted with E2E keys. See Section 7 for a
related item in the To Do list.

4.4. HBH Keys and Hop Operations

To ensure the integrity of transmitted media packets, this framework
requires that every packet be authenticated hop-by-hop (HBH) between
an endpoint and a Media Distributor, as well between Media
Distributors. The authentication key used for hop-by-hop
authentication is derived from an SRTP master key shared only on the
respective hop (HBH Key(j); j={a given hop}). Each HBH Key(j) is
distinct per hop and no two hops ever intentionally use the same SRTP
master key.

Using hop-by-hop authentication gives the Media Distributor the
ability to change certain RTP header values. Which values the Media

Distributor can change in the RTP header are defined in
[I-D.ietf-perc-double]. RTCP can only be encrypted, giving the Media
Distributor the flexibility to forward RTCP content unchanged,
transmit compound RTCP packets or to initiate RTCP packets for
reporting statistics or conveying other information. Performing hop-
by-hop authentication for all RTP and RTCP packets also helps provide
replay protection (see Section 6).

If there is a need to encrypt one or more RTP header extensions hop-
by-hop, an encryption key is derived from the hop-by-hop SRTP master
key to encrypt header extensions as per [RFC6904]. This will still
give the Media Distributor visibility into header extensions, such as
the one used to determine audio level [RFC6464] of conference
participants. Note that when RTP header extensions are encrypted,
all hops - in the untrusted domain at least - will need to decrypt
and re-encrypt these encrypted header extensions.

4.5. Key Exchange

To facilitate key exchange required to establish or generate an E2E
key and a HBH key for an endpoint and the same HBH key for the Media
Distributor, this framework utilizes a DTLS-SRTP [RFC5764]
association between an endpoint and the Key Distributor. To
establish this association, an endpoint will send DTLS-SRTP messages
to the Media Distributor which will then forward them to the Media
Distributor as defined in [I-D.jones-perc-dtls-tunnel]. The Key
Encryption Key (KEK) (i.e., EKT Key) is also conveyed by the Key
Distributor over the DTLS association to endpoints via procedures
defined in PERC EKT [I-D.ietf-perc-srtp-ekt-diet].

Media Distributors use DTLS-SRTP [RFC5764] directly with a peer Media
Distributor to establish HBH keys for transmitting RTP and RTCP
packets that peer Media Distributor. The Key Distributor does not
facilitate establishing HBH keys for use between Media Distributors.

4.5.1. Initial Key Exchange and Key Distributor

The procedures defined in DTLS Tunnel for PERC
[I-D.jones-perc-dtls-tunnel] establish one or more DTLS tunnels
between the Media Distributor and Key Distributor, making it is
possible for the Media Distributor to facilitate the establishment of
a secure DTLS association between each endpoint and the Key
Distributor as shown the following figure. The DTLS association
between endpoints and the Key Distributor will enable each endpoint
to receive E2E key information, Key Encryption Key (KEK) information
(i.e., EKT Key), and HBH key information. At the same time, the Key
Distributor can securely provide the HBH key information to the Media
Distributor. The key information summarized here may include the

SRTP master key, SRTP master salt, and the negotiated cryptographic
transform.

```
                        +-----------+
              KEK info |    Key    | HBH Key info to
           to Endpoints |Distributor| Endpoints & Media Distributor
                        +-----------+
                          # ^ ^ #
                          # | | #-DTLS Tunnel
                          # | | #
+-----------+           +-----------+            +-----------+
| Endpoint  |   DTLS    |   Media   |   DTLS     | Endpoint  |
|    KEK    |<----------|Distributor|----------->|    KEK    |
| HBH Key(j)| to Key Dist | HBH Keys  | to Key Dist | HBH Key(j)|
+-----------+           +-----------+            +-----------+
```
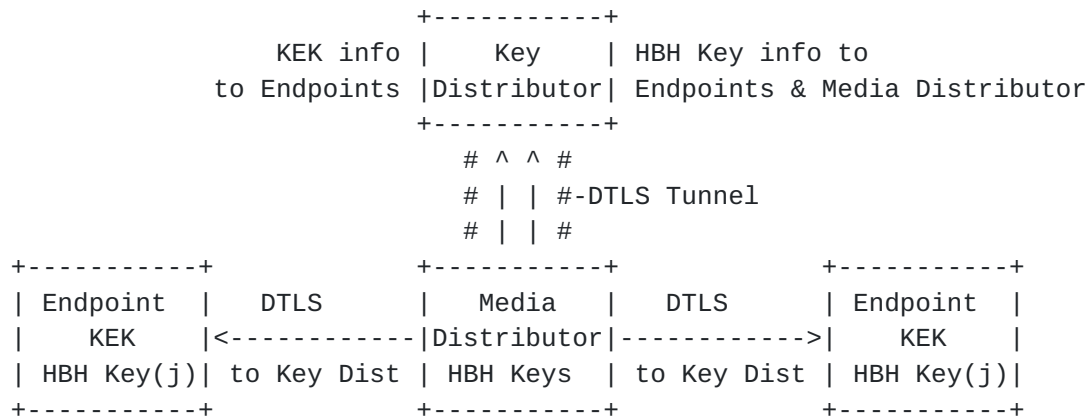
Figure 3: Exchanging Key Information Between Entities

Endpoints will establish a DTLS-SRTP association over the RTP
session's media ports for the purposes of key information exchange
with the Key Distributor. The Media Distributor will not terminate
the DTLS signaling, but will instead forward DTLS packets received
from an endpoint on to the Key Distributor (and vice versa) via a
tunnel established between Media Distributor and the Key Distributor.
This tunnel used to encapsulate the DTLS-SRTP signaling between the
Key Distributor and endpoints will also be used to convey HBH key
information from the Key Distributor to the Media Distributor, so no
additional protocol or interface is required.

4.5.2. Key Exchange during a Conference

Following the initial key information exchange with the Key
Distributor, endpoints will be able to encrypt media end-to-end with
their E2E Key(i), sending that E2E Key(i) to other endpoints
encrypted with KEK, and will be able to encrypt and authenticate RTP
packets using local HBH Key(j). The procedures defined do not allow
the Media Distributor to gain access to the KEK information,
preventing it from gaining access to any endpoint's E2E key and
subsequently decrypting media.

The KEK (i.e., EKT Key) may need to change from time-to-time during
the life of a conference, such as when a new participant joins or
leaves a conference. Dictating if, when or how often a conference is
to be re-keyed is outside the scope of this document, but this
framework does accommodate re-keying during the life of a conference.

When a Key Distributor decides to rekey a conference, it transmits a
specific message defined in PERC EKT [I-D.ietf-perc-srtp-ekt-diet] to
each of the conference participants. The endpoint MUST create a new
SRTP master key and prepare to send that key inside a Full EKT Field
using the new EKT Key. Since it may take some time for all of the
endpoints in conference to finish re-keying, senders MUST delay a
short period of time before sending media encrypted with the new
master key, but it MUST be prepared to make use of the information
from a new inbound EKT Key immediately. See Section 2.2.2 of
[I-D.ietf-perc-srtp-ekt-diet].

5. Entity Trust

It is important to this solution framework that the entities can
trust and validate the authenticity of other entities, especially the
Key Distributor and endpoints. The details of this are outside the
scope of specification but a few possibilities are discussed in the
following sections. The key requirements is that endpoints can
verify they are connected to the correct Key Distributor for the
conference and the Key Distributor can verify the endpoints are the
correct endpoints for the conference.

Two possible approaches to solve this are Identity Assertions and
Certificate Fingerprints.

5.1. Identity Assertions

WebRTC Identity assertion (EDITOR NOTE: add I-D reference) can be
used to bind the identity of the user of the endpoint to the
fingerprint of the DTLS-SRTP certificate used for the call. This
certificate is unique for a given call and a conference. This allows
the Key Distributor to ensure that only authorized users participate
in the conference. Similarly the Key Distributor can create a WeBRTC
Identity assertion bound the fingerprint of the unique certificate
used by the Key Distributor for this conference so that the endpoint
can validate it is talking to the correct Key Distributor.

5.2. Certificate Fingerprints in Session Signaling

Entities managing session signaling are generally assumed to be
untrusted in the PERC framework. However, there are some deployment
scenarios where parts of the session signaling may be assumed
trustworthy for the purposes of exchanging, in a manner that can be
authenticated, the fingerprint of an entity's certificate.

As a concrete example, SIP [RFC3261] and SDP [RFC4566] can be used to
convey the fingerprint information per [RFC5763]. An endpoint's SIP
User Agent would send an INVITE message containing SDP for the media

session along with the endpoint's certificate fingerprint, which can
be signed using the procedures described in [RFC4474] for the benefit
of forwarding the message to other entities. Other entities can now
verify the fingerprints match the certificates found in the DTLS-SRTP
connections to find the identity of the far end of the DTLS-SRTP
connection and check that is the authorized entity.

Ultimately, if using session signaling, an endpoint's certificate
fingerprint would need to be securely mapped to a user and conveyed
to the Key Distributor so that it can check that that user is
authorized. Similarly, the Key Distributor's certificate fingerprint
can be conveyed to endpoint in a manner that can be authenticated as
being an authorized Key Distributor for this conference.

6. Attacks on Privacy Enhanced RTP Conferencing

This framework, and the individual protocols defined to support it,
must take care to not increase the exposure to Denial of Service
(DoS) attacks by untrusted or third-party entities and should take
measures to mitigate, where possible, more serious DoS attacks from
on-path and off-path attackers.

The following section enumerates the kind of attacks that will be
considered in the development of this framework's solution.

6.1. Third Party Attacks

On-path attacks are mitigated by HBH integrity protection and
encryption. The integrity protection mitigates packet modification
and encryption makes selective blocking of packets harder, but not
impossible.

Off-path attackers may try connecting to different PERC entities and
send specifically crafted packets. A successful attacker might be
able to get the Media Distributor to forward such packets. If not
making use of HBH authentication on the Media Distributor, such an
attack could only be detected in the receiving endpoints where the
forged packets would finally be dropped.

Another potential attack is a third party claiming to be a Media
Distributor, fooling endpoints in to sending packets to the false
Media Distributor instead of the correct one. The deceived sending
endpoints could incorrectly assuming their packets have been
delivered to endpoints when they in fact have not. Further, the
false Media Distributor may cascade to another legitimate Media
Distributor creating a false version of the real conference.

This attack can be mitigated by the false Media Distributor not being
authenticated by the Key Distributor during PERC Tunnel
establishment. Without the tunnel in place, endpoints will not
establish secure associations with the Key Distributor and receive
the KEK, causing the conference to not proceed.

6.2. Media Distributor Attacks

The Media Distributor can attack the session in a number of possible
ways.

6.2.1. Denial of service

Any modification of the end-to-end authenticated data will result in
the receiving endpoint getting an integrity failure when performing
authentication on the received packet.

The Media Distributor can also attempt to perform resource
consumption attacks on the receiving endpoint. One such attack would
be to insert random SSRC/CSRC values in any RTP packet with an inband
key-distribution message attached (i.e., Full EKT Field). Since such
a message would trigger the receiver to form a new cryptographic
context, the Media Distributor can attempt to consume the receiving
endpoints resources.

Another denial of service attack is where the Media Distributor
rewrites the PT field to indicate a different codec. The effect of
this attack is that any payload packetized and encoded according to
one RTP payload format is then processed using another payload format
and codec. Assuming that the implementation is robust to random
input, it is unlikely to cause crashes in the receiving software/
hardware. However, it is not unlikely that such rewriting will cause
severe media degradation.

For audio formats, this attack is likely to cause highly disturbing
audio and/or can be damaging to hearing and playout equipment.

6.2.2. Replay Attack

Replay attack is when an already received packets from a previous
point in the RTP stream is replayed as new packet. This could, for
example, allow a Media Distributor to transmit a sequence of packets
identified as a user saying "yes", instead of the "no" the user
actually said.

The mitigation for a replay attack is to prevent old packets beyond a
small-to-modest jitter and network re-ordering sized window to be

rejected. End-to-end replay protection MUST be provided for the
whole duration of the conference.

6.2.3. Delayed Playout Attack

The delayed playout attack is a variant of the replay attack. This
attack is possible even if E2E replay protection is in place.
However, due to fact that the Media Distributor is allowed to select
a sub-set of streams and not forward the rest to a receiver, such as
in forwarding only the most active speakers, the receiver has to
accept gaps in the E2E packet sequence. The issue with this is that
a Media Distributor can select to not deliver a particular stream for
a while.

Within the window from last packet forwarded to the receiver and the
latest received by the Media Distributor, the Media Distributor can
select an arbitrary starting point when resuming forwarding packets.
Thus what the media source said can be substantially delayed at the
receiver with the receiver believing that it is what was said just
now, and only delayed due to transport delay.

6.2.4. Splicing Attack

The splicing attack is an attack where a Media Distributor receiving
multiple media sources splices one media stream into the other. If
the Media Distributor is able to change the SSRC without the receiver
having any method for verifying the original source ID, then the
Media Distributor could first deliver stream A and then later forward
stream B under the same SSRC as stream A was previously using. Not
allowing the Media Distributor to change the SSRC mitigates this
attack.

7. To-Do List

   o The mapping of endpoints-to-conference identifiers may need to be
      conveyed in the framework. Need Revisit this text after a design
      choice is made between alternatives.

   o Consider adding a list of RTP header extensions that should/must
      not be E2E encrypted.

   o Endpoints, Key Distributor and Media Distributor provider must
      securely convey their respective certificate information directly
      or indirectly via some means, such as via an identity service
      provider.

   o If as in "Double" draft, the ROC value is no longer in the clear
      and associated with the "outer" protection scheme, we may need to

require that the Media Distributor maintain a separate ROC value
for each SSRC sent to each endpoint. This ROC value should start
at 0 regardless of the sequence number in that first packet sent
to an endpoint.

o Investigate adding ability to enable the transmission of one-way
media from a non-trusted device (e.g., announcements). One
possible solution is to have the Key Distributor send an "ekt_key"
message that is explicitly labeled for receive-only and giving
that to announcement servers. A possible approach would be to
define EKT Keys with a SPI > 32000, for example, for this purpose
and endpoints should only use those EKT Keys to decrypt Full EKT
Fields received from such transmitters. Endpoints would never
send media with EKT Keys with those SPI values.

8. IANA Considerations

There are no IANA considerations for this document.

9. Security Considerations

[TBD]

10. Acknowledgments

The authors would like to thank Mo Zanaty and Christian Oien for
invaluable input on this document. Also, we would like to
acknowledge Nermeen Ismail for serving on the initial versions of
this document as a co-author.

11. References

11.1. Normative References

[I-D.ietf-perc-double]
          Jennings, C., Jones, P., and A. Roach, "SRTP Double
          Encryption Procedures", draft-ietf-perc-double-00 (work in
          progress), May 2016.

[I-D.ietf-perc-srtp-ekt-diet]
          Mattsson, J., McGrew, D., Wing, D., and C. Jennings,
          "Encrypted Key Transport for Secure RTP", draft-ietf-perc-
          srtp-ekt-diet-00 (work in progress), May 2016.

[I-D.jones-perc-dtls-tunnel]
          Jones, P., "DTLS Tunnel between Media Distribution Device
          and Key Management Function to Facilitate Key Exchange",
          draft-jones-perc-dtls-tunnel-03 (work in progress), July
          2016.

[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>.

[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, <http://www.rfc-editor.org/info/rfc3550>.

[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,
          <http://www.rfc-editor.org/info/rfc3711>.

[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,
          <http://www.rfc-editor.org/info/rfc5764>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure
          Real-time Transport Protocol (SRTP)", RFC 6904, DOI
          10.17487/RFC6904, April 2013,
          <http://www.rfc-editor.org/info/rfc6904>.

11.2. Informative References

[I-D.ietf-avtcore-rtp-topologies-update]
          Westerlund, M. and S. Wenger, "RTP Topologies", draft-
          ietf-avtcore-rtp-topologies-update-10 (work in progress),
          July 2015.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
          A., Peterson, J., Sparks, R., Handley, M., and E.
          Schooler, "SIP: Session Initiation Protocol", RFC 3261,
          DOI 10.17487/RFC3261, June 2002,
          <http://www.rfc-editor.org/info/rfc3261>.

   [RFC4474] Peterson, J. and C. Jennings, "Enhancements for
              Authenticated Identity Management in the Session
              Initiation Protocol (SIP)", RFC 4474, DOI 10.17487/
              RFC4474, August 2006,
              <http://www.rfc-editor.org/info/rfc4474>.

   [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
              Description Protocol", RFC 4566, DOI 10.17487/RFC4566,
              July 2006, <http://www.rfc-editor.org/info/rfc4566>.

   [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework
              for Establishing a Secure Real-time Transport Protocol
              (SRTP) Security Context Using Datagram Transport Layer
              Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May
              2010, <http://www.rfc-editor.org/info/rfc5763>.

   [RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time
              Transport Protocol (RTP) Header Extension for Client-to-
              Mixer Audio Level Indication", RFC 6464, DOI 10.17487/
              RFC6464, December 2011,
              <http://www.rfc-editor.org/info/rfc6464>.

Authors' Addresses

   Paul E. Jones
   Cisco
   7025 Kit Creek Rd.
   Research Triangle Park, North Carolina 27709
   USA

   Phone: +1 919 476 2048
   Email: paulej@packetizer.com

   David Benham
   Cisco
   170 West Tasman Drive
   San Jose, California 95134
   USA

   Email: dbenham@cisco.com

Christian Groves
Huawei
Melbourne
Australia

Email: Christian.Groves@nteczone.com