Stream: Internet Engineering Task Force (IETF)
RFC: 8871
Category: Standards Track
Published: October 2020
ISSN: 2070-1721
Authors: P. Jones *Cisco* D. Benham *Independent* C. Groves *Independent*

RFC 8871
A Solution Framework for Private Media in Privacy-Enhanced RTP Conferencing (PERC)

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 Distributors are not trusted with the end-to-end media encryption keys. The solution builds upon existing security mechanisms defined for the Real-time Transport Protocol (RTP).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
2. Conventions Used in This Document
3. PERC Entities and Trust Model
   3.1. Untrusted Entities
      3.1.1. Media Distributor
      3.1.2. Call Processing
   3.2. Trusted Entities
      3.2.1. Endpoint
      3.2.2. Key Distributor
4. Framework for PERC
   4.1. E2E-Authenticated and HBH-Authenticated Encryption
   4.2. E2E Key Confidentiality
   4.3. E2E Keys and Endpoint Operations
   4.4. HBH Keys and Per-Hop Operations
   4.5. Key Exchange
      4.5.1. Initial Key Exchange and Key Distributor
      4.5.2. Key Exchange during a Conference
5. Authentication
   5.1. Identity Assertions
   5.2. Certificate Fingerprints in Session Signaling
   5.3. Conference Identification
6. PERC Keys
   6.1. Key Inventory and Management Considerations
   6.2. DTLS-SRTP Exchange Yields HBH Keys
   6.3. The Key Distributor Transmits the KEK (EKT Key)
   6.4. Endpoints Fabricate an SRTP Master Key
   6.5. Summary of Key Types and Entity Possession

7. Encrypted Media Packet Format
8. Security Considerations
8.1. Third-Party Attacks
8.2. Media Distributor Attacks
8.2.1. Denial of Service
8.2.2. Replay Attacks
8.2.3. Delayed Playout Attacks
8.2.4. Splicing Attacks
8.2.5. RTCP Attacks
8.3. Key Distributor Attacks
8.4. Endpoint Attacks
9. IANA Considerations
10. References
10.1. Normative References
10.2. Informative References
Acknowledgments
Authors' Addresses

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, translated, 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 Media Distributors to each of the other participants. Media Distributors often forward only a subset of flows based on voice activity detection or other criteria. In some instances, Media Distributors may make limited modifications to RTP headers [RFC3550], 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. Virtualized public cloud environments have been viewed as less secure, since resources are not always physically controlled by those who use them. This document defines improved security so as to lower the barrier to taking advantage of those environments.

This document defines a solution framework wherein media privacy is ensured by making it impossible for a Media Distributor 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 RTP Control Protocol (RTCP) packets [RFC3550]. The framework also prevents replay attacks by authenticating each packet transmitted between a given participant and the Media Distributor using a unique key per endpoint that is independent from the key for media encryption and authentication.

This solution framework provides 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**", "**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.

Additionally, this solution framework uses the following terms and abbreviations:

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

Hop-by-Hop (HBH): Communications between an endpoint and a Media Distributor or between Media Distributors.

Trusted Endpoint (or simply 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 devices, and more, that are in the trusted domain for a given deployment. In the context of WebRTC [W3C.CR-webrtc-20191213], where control of a session is divided between a JavaScript application and a browser, the browser acts as the Trusted Endpoint for purposes of this framework (just as it acts as the endpoint for DTLS-SRTP [RFC5764] in one-to-one calls).

Media Distributor (MD): An RTP middlebox that forwards endpoint media content (e.g., voice or video data) unaltered -- either a subset or all of the flows at any given time -- and is never allowed to have access to E2E encryption keys. It operates according to the Selective Forwarding Middlebox RTP topologies [RFC7667] per the constraints defined by the Private Media in Privacy-Enhanced RTP Conferencing (PERC) system, which includes, but is not limited to, having no access to RTP media plaintext and having limits on what RTP header fields it can alter. The header fields that may be modified by a Media Distributor are enumerated in Section 4 of the double cryptographic transform specification [RFC8723] and chosen with respect to utility and the security considerations outlined in this document.

Key Distributor: An entity that is a logical function that distributes keying material and related information to Trusted Endpoints and Media Distributor(s) -- only what 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 Distributors.

Call Processing: All Trusted Endpoints connect to a conference via a call processing dialog, e.g., with the "focus" as defined in "A Framework for Conferencing with the Session Initiation Protocol (SIP)" [RFC4353].

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

Figure 1 depicts the trust relationships, direct or indirect, between entities described in the subsequent subsections. 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 it keeps the definitions and examples focused by only using 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 that the privacy of the participant's media is not compromised. The conferencing infrastructure in such a domain is still trusted with reliably connecting the participants together in a conference but is not trusted with keying material needed to decrypt any of the participant's media. Entities in such less-trustworthy domains are referred to as untrusted entities from this point forward.

It is important to understand that "untrusted" in this document does not mean that an entity is not expected to function properly. Rather, it means only that the entity does not have access to the E2E media encryption keys.

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 also relays secured messaging between the endpoints and the Key Distributor and acquires per-hop key information from the Key Distributor. The actual media content must not be decryptable by a Media Distributor, as it is not trusted to have access to the E2E media encryption keys. The key exchange mechanisms specified in this framework prevent the Media Distributor from gaining access to the E2E media encryption keys.

An endpoint's ability to connect to a conference serviced by a Media Distributor implies that the endpoint is authorized to have access to the E2E media encryption keys, although the Media Distributor does not have the ability to determine whether an endpoint is authorized. Instead, the Key Distributor is responsible for authenticating the endpoint (e.g., using WebRTC Identity [RFC8827]) and determining its authorization to receive E2E and HBH media encryption keys.

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 8) to a level equal to or better than traditional conferencing (non-PERC) deployments.

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

3.1.2. Call Processing

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

Call processing 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 of subsequently joining a conference hosted through one or more Media Distributors. Call processing may optionally ensure the privacy of call signaling messages between itself (call processing), the endpoint, and other entities.

3.2. Trusted Entities

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

3.2.1. Endpoint

An endpoint is considered trusted and has 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 take measures to mitigate the adverse effects of denial-of-service attacks (refer to Section 8) from other entities, including from other endpoints, to a level equal to or better than traditional conference (non-PERC) deployments.

3.2.2. Key Distributor

The Key Distributor, which may be co-located with an endpoint or exist standalone, is responsible for providing key information to endpoints for both E2E and HBH security and for providing key information to Media Distributors for HBH security.

Interaction between the Key Distributor and call processing is necessary for proper conference-to-endpoint mappings. This is described in Section 5.3.

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

The Key Distributor needs to know which endpoints and which Media Distributors are authorized to participate in the conference. How the Key Distributor obtains this information is outside the scope of this document. However, Key Distributors **MUST** reject DTLS associations with any unauthorized endpoint or Media Distributor.

4. Framework for PERC

The purpose of this framework is to define a means through which media privacy is ensured when communicating within a conferencing environment consisting of one or more Media Distributors that only switch, and hence do 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 the Secure Real-time Transport Protocol (SRTP) [RFC3711], Encrypted Key Transport (EKT) [RFC8870], and DTLS-SRTP.

4.1. E2E-Authenticated and HBH-Authenticated Encryption

This solution framework focuses on the E2E privacy and integrity of the participant's media by limiting access to only trusted entities to the E2E key used for authenticated E2E encryption. However, this framework does give a Media Distributor access to RTP header fields and header extensions, as well as the ability to modify a certain subset of the header fields and to add or change 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 (a distinct E2E key for each transmitted media flow) for authenticated encryption of RTP media between endpoints and an "outer" key (a HBH key) known only to a Media Distributor or the adjacent endpoint for the hop between an endpoint and a Media Distributor or peer endpoint. An endpoint will receive one or more E2E keys from every other endpoint in the conference that correspond to the media flows transmitted by those other endpoints, while HBH keys are derived from the DTLS-SRTP association with the Key Distributor. Two communicating Media Distributors use DTLS-SRTP associations directly with each other to obtain the HBH keys they will use. See Section 4.5 for more details on key exchange.

```
+-------------+                                +-------------+
|             |################################|             |
|    Media    |------------------------ *----->|    Media    |
| Distributor |<----------------------*-|------| Distributor |
|      X      |#####################*#|#|######|      Y      |
|             |                     | | |      |             |
+-------------+                     | | |      +-------------+
   #  ^ |  #          HBH Key (XY) -+ | |         #  ^ |  #
   #  | |  #           E2E Key (B) ---+ |         #  | |  #
   #  | |  #           E2E Key (A) -----+         #  | |  #
   #  | |  #                                      #  | |  #
   #  | |  #                                      #  | |  #
   #  | |  *---- HBH Key (AX)    HBH Key (YB) ----*  | |  #
   #  | |  #                                      #  | |  #
   #  *--------- E2E Key (A)      E2E Key (A) ---------*  #
   #  | *------- E2E Key (B)      E2E Key (B) -------* |  #
   #  | |  #                                      #  | |  #
   #  | v  #                                      #  | v  #
+-------------+                                +-------------+
| Endpoint A  |                                | Endpoint B  |
+-------------+                                +-------------+
```

Figure 2: E2E and HBH Keys Used for Authenticated Encryption of SRTP Packets

The double transform [RFC8723] enables endpoints to perform encryption using both the E2E and HBH contexts while still preserving the same overall interface as other SRTP transforms. The Media Distributor simply uses the corresponding normal (single) AES-GCM transform, keyed with the appropriate HBH keys. See Section 6.1 for a description of the keys used in PERC and Section 7 for a diagram of how encrypted RTP packets appear on the wire.

RTCP is only encrypted hop by hop -- not end to end. This framework does not provide an additional step for RTCP-authenticated encryption. Rather, implementations utilize the existing procedures specified in [RFC3711]; those procedures use the same outer, HBH cryptographic context chosen in the double transform operation described above. For this reason, endpoints **MUST NOT** send confidential information via RTCP.

4.2. E2E Key Confidentiality

To ensure the confidentiality of E2E keys shared between endpoints, endpoints use a common Key Encryption Key (KEK) that is known only by the trusted entities in a conference. That KEK, defined in the EKT specification [RFC8870] as the EKT Key, is used to subsequently encrypt the SRTP master key used for E2E-authenticated encryption of media sent by a given endpoint. Each endpoint in the conference creates an SRTP master key for E2E-authenticated encryption and keeps track of the E2E keys received via the Full EKT Tag for each distinct synchronization source (SSRC) in the conference so that it can properly decrypt received media. An endpoint may change its E2E key at any time and advertise that new key to the conference as specified in [RFC8870].

4.3. E2E Keys and Endpoint Operations

Any given RTP media flow is identified by its SSRC, and an endpoint might send more than one at a time and change the mix of media flows transmitted during the lifetime of a conference.

Thus, an endpoint **MUST** maintain a list of SSRCs from received RTP flows and each SSRC's associated E2E key information. An endpoint **MUST** discard old E2E keys no later than when it leaves the conference.

If the packet is to contain RTP header extensions, it should be noted that those extensions are only encrypted hop by hop per [RFC8723]. For this reason, endpoints **MUST NOT** transmit confidential information via RTP header extensions.

4.4. HBH Keys and Per-Hop Operations

To ensure the integrity of transmitted media packets, it is **REQUIRED** that every packet be authenticated hop by hop between an endpoint and a Media Distributor, as well as between Media Distributors. The authentication key used for HBH authentication is derived from an SRTP master key shared only on the respective hop. Each HBH key is distinct per hop, and no two hops ever use the same SRTP master key.

While endpoints also perform HBH authentication, the ability of the endpoints to reconstruct the original RTP header also enables the endpoints to authenticate RTP packets end to end. This design yields flexibility to the Media Distributor to change certain RTP header values as packets are forwarded. Values that the Media Distributor can change in the RTP header are defined in [RFC8723]. RTCP can only be encrypted hop by hop, giving the Media Distributor the flexibility to (1) forward RTCP content unchanged, (2) transmit compound RTCP packets, (3) initiate RTCP packets for reporting statistics, or (4) convey other information. Performing HBH authentication for all RTP and RTCP packets also helps provide replay protection (see Section 8). The use of the replay protection mechanism specified in Section 3.3.2 of [RFC3711] is **REQUIRED** at each hop.

If there is a need to encrypt one or more RTP header extensions hop by hop, the endpoint derives an encryption key from the HBH SRTP master key to encrypt header extensions as per [RFC6904]. This still gives the Media Distributor visibility into header extensions, such as the one used to determine the audio level [RFC6464] of conference participants. Note that when RTP header extensions are encrypted, all hops need to decrypt and re-encrypt these encrypted header extensions. Please refer to Sections 5.1, 5.2, and 5.3 of [RFC8723] for procedures to perform RTP header extension encryption and decryption.

4.5. Key Exchange

In brief, the keys used by any given endpoints are determined as follows:

- The HBH keys that the endpoint uses to send and receive SRTP media are derived from a DTLS handshake that the endpoint performs with the Key Distributor (following normal DTLS-SRTP procedures).
- The E2E key that an endpoint uses to send SRTP media can be either set from the DTLS-SRTP association with the Key Distributor or chosen by the endpoint. It is then distributed to other endpoints in a Full EKT Tag, encrypted under an EKT Key provided to the client by the Key Distributor within the DTLS channel they negotiated. Note that an endpoint **MAY** create a different E2E key per media flow, where a media flow is identified by its SSRC value.
- Each E2E key that an endpoint uses to receive SRTP media is set by receiving a Full EKT Tag from another endpoint.
- The HBH keys used between two Media Distributors are derived via DTLS-SRTP procedures employed directly between them.

4.5.1. Initial Key Exchange and Key Distributor

The Media Distributor maintains a tunnel with the Key Distributor (e.g., using the tunnel protocol defined in [PERC-DTLS]), making it possible for the Media Distributor to facilitate the establishment of a secure DTLS association between each endpoint and the Key Distributor as shown in Figure 3. The DTLS association between endpoints and the Key Distributor enables each endpoint to generate E2E and HBH keys and receive the KEK. At the same time, the Key Distributor securely provides the HBH key information to the Media Distributor. The key information summarized here may include the SRTP master key, the SRTP master salt, and the negotiated cryptographic transform.

```
                               +-----------+
                      KEK info |    Key    | HBH Key info to
                  to Endpoints |Distributor| Endpoints & Media Distributor
                               +-----------+
                                  # ^ ^ #
                                  # | | #--- Tunnel
                                  # | | #
      +-----------+            +-----------+            +-----------+
      | Endpoint  |   DTLS     |   Media   |   DTLS     | Endpoint  |
      |    KEK    |<------------|Distributor|------------>|    KEK    |
      |  HBH Key  | to Key Dist | HBH Keys  | to Key Dist |  HBH Key  |
      +-----------+            +-----------+            +-----------+
```

Figure 3: Exchanging Key Information between Entities

In addition to the secure tunnel between the Media Distributor and the Key Distributor, there are two additional types of security associations utilized as a part of the key exchange, as discussed in the following paragraphs. One is a DTLS-SRTP association between an endpoint and the Key Distributor (with packets passing through the Media Distributor), and the other is a DTLS-SRTP association between peer Media Distributors.

Endpoints establish a DTLS-SRTP association over the RTP session with the Media Distributor and its media ports for the purposes of key information exchange with the Key Distributor. The Media Distributor does not terminate the DTLS signaling but instead forwards DTLS packets received from an endpoint on to the Key Distributor (and vice versa) via a tunnel established between the Media Distributor and the Key Distributor.

When establishing the DTLS association between endpoints and the Key Distributor, the endpoint **MUST** act as the DTLS client, and the Key Distributor **MUST** act as the DTLS server. The KEK is conveyed by the Key Distributor over the DTLS association to endpoints via procedures defined in EKT [RFC8870] via the EKTKey message.

The Key Distributor **MUST NOT** establish DTLS-SRTP associations with endpoints without first authenticating the Media Distributor tunneling the DTLS-SRTP packets from the endpoint.

Note that following DTLS-SRTP procedures for the cipher defined in [RFC8723], the endpoint generates both E2E and HBH encryption keys and salt values. Endpoints **MUST** either use the DTLS-SRTP-generated E2E key for transmission or generate a fresh E2E key. In either case, the generated SRTP master salt for E2E encryption **MUST** be replaced with the salt value provided by the Key Distributor via the EKTKey message. That is because every endpoint in the conference uses the same SRTP master salt. The endpoint only transmits the SRTP master key (not the salt) used for E2E encryption to other endpoints in RTP/RTCP packets per [RFC8870].

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

4.5.2. Key Exchange during a Conference

Following the initial key information exchange with the Key Distributor, an endpoint is able to encrypt media end to end with an E2E key, sending that E2E key to other endpoints encrypted with the KEK, and is able to encrypt and authenticate RTP packets using a HBH key. This framework does 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 may need to change from time to time during the lifetime 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 rekeyed is outside the scope of this document, but this framework does accommodate rekeying during the lifetime of a conference.

When a Key Distributor decides to rekey a conference, it transmits a new EKTKey message containing the new EKT Key to each of the conference participants. Upon receipt of the new EKT Key, the endpoint **MUST** create a new SRTP master key and prepare to send that key inside a FullEKTField using the new EKT Key as per Section 4.5 of [RFC8870]. In order to allow time for all endpoints in the conference to receive the new keys, the sender should follow the recommendations in Section 4.6 of [RFC8870]. On receiving a new EKT Key, endpoints **MUST** be prepared to decrypt EKT tags using the new key. The EKT Security Parameter Index (SPI) field is used to differentiate between EKT Tags encrypted with the old and new keys.

After rekeying, an endpoint **SHOULD** retain prior SRTP master keys and EKT Keys for a period of time sufficient for the purpose of ensuring that it can decrypt late-arriving or out-of-order packets or packets sent by other endpoints that used the prior keys for a period of time after rekeying began. An endpoint **MAY** retain old keys until the end of the conference.

Endpoints **MAY** follow the procedures in Section 5.2 of [RFC5764] to renegotiate HBH keys as desired. If new HBH keys are generated, the new keys are also delivered to the Media Distributor following the procedures defined in [PERC-DTLS] as one possible method.

At any time, endpoints **MAY** change the E2E encryption key being used. An endpoint **MUST** generate a new E2E encryption key whenever it receives a new EKT Key. After switching to a new key, the new key is conveyed to other endpoints in the conference in RTP/RTCP packets per [RFC8870].

5. Authentication

It is important that entities can validate the authenticity of other entities, especially the Key Distributor and endpoints. Details on this topic are outside the scope of this specification, but a few possibilities are discussed in the following sections. The critical requirements are that (1) an endpoint can verify that it is connected to the correct Key Distributor for the conference and (2) the Key Distributor can verify that the endpoint is the correct endpoint for the conference.

Two possible approaches to resolve this situation are identity assertions and certificate fingerprints.

5.1. Identity Assertions

A WebRTC identity assertion [RFC8827] is 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 certificate is unique for a given call and a conference, allowing the Key Distributor to ensure that only authorized users participate in the conference. Similarly, the Key Distributor can create a WebRTC identity assertion to bind the fingerprint of the unique certificate used by the Key Distributor for this conference so that the endpoint can verify that it is talking to the correct Key Distributor. Such a setup requires an Identity Provider (IdP) trusted by the endpoints and the 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 the Session Description Protocol (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 [RFC8224] for the benefit of forwarding the message to other entities by the focus [RFC4353]. Other entities can verify that 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 verify that it 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 the user in question is authorized. Similarly, the Key Distributor's certificate fingerprint can be conveyed to an endpoint in a manner that can be authenticated as being an authorized Key Distributor for this conference.

5.3. Conference Identification

The Key Distributor needs to know what endpoints are being added to a given conference. Thus, the Key Distributor and the Media Distributor need to know endpoint-to-conference mappings, which are enabled by exchanging a conference-specific unique identifier as described in [PERC-DTLS]. How this unique identifier is assigned is outside the scope of this document.

6. PERC Keys

This section describes the various keys employed by PERC.

6.1. Key Inventory and Management Considerations

This section summarizes the several different keys used in the PERC framework, how they are generated, and what purpose they serve.

The keys are described in the order in which they would typically be acquired.

The various keys used in PERC are shown in Table 1 below.

Key	Description
HBH Key	SRTP master key used to encrypt media hop by hop.
KEK (EKT Key)	Key shared by all endpoints and used to encrypt each endpoint's E2E SRTP master key so receiving endpoints can decrypt media.
E2E Key	SRTP master key used to encrypt media end to end.

Table 1: Key Inventory

While the number of key types is very small, it should be understood that the actual number of distinct keys can be large as the conference grows in size.

As an example, with 1,000 participants in a conference, there would be at least 1,000 distinct SRTP master keys, all of which share the same master salt. Each of those keys is passed through the Key Derivation Function (KDF) as defined in [RFC3711] to produce the actual encryption and authentication keys.

Complicating key management is the fact that the KEK can change and, when it does, the endpoints generate new SRTP master keys that are associated with a new EKT SPI. Endpoints might retain old keys for a period of time to ensure that they can properly decrypt late-arriving or out-of-order packets, which means that the number of keys held during that period of time might be substantially higher.

A more detailed explanation of each of the keys follows.

6.2. DTLS-SRTP Exchange Yields HBH Keys

The first set of keys acquired are for HBH encryption and decryption. Per the double transform procedures [RFC8723], the endpoint performs a DTLS-SRTP exchange with the Key Distributor and receives a key that is, in fact, "double" the size that is needed. The E2E part is the first half of the key, so the endpoint discards that information when generating its own key. The second half of the keying material is for HBH operations, so that half of the key (corresponding to the least significant bits) is assigned internally as the HBH key.

The Key Distributor informs the Media Distributor of the HBH key. Specifically, the Key Distributor sends the least significant bits corresponding to the half of the keying material determined through DTLS-SRTP with the endpoint to the Media Distributor. A salt value is generated along with the HBH key. The salt is also longer than needed for HBH operations; thus, only the least significant bits of the required length (half of the generated salt material) are sent to the Media Distributor. One way to transmit this key and salt information is via the tunnel protocol defined in [PERC-DTLS].

No two endpoints have the same HBH key; thus, the Media Distributor **MUST** keep track of each distinct HBH key (and the corresponding salt) and use it only for the specified hop.

The HBH key is also used for HBH encryption of RTCP. RTCP is not E2E-encrypted in PERC.

6.3. The Key Distributor Transmits the KEK (EKT Key)

The Key Distributor sends the KEK (the EKT Key per [RFC8870]) to the endpoint via the aforementioned DTLS-SRTP association. This key is known only to the Key Distributor and endpoints; it is the most important entity to protect, since having knowledge of this key (and the SRTP master salt transmitted as a part of the same message) allows an entity to decrypt any media packet in the conference.

Note that the Key Distributor can send any number of EKT Keys to endpoints. This information is used to rekey the entire conference. Each key is identified by an SPI value. Endpoints **MUST** expect that a conference might be rekeyed when a new participant joins a conference or when a participant leaves a conference, in order to protect the confidentiality of the conversation before and after such events.

The SRTP master salt to be used by the endpoint is transmitted along with the EKT Key. All endpoints in the conference utilize the same SRTP master salt that corresponds with a given EKT Key.

The Full EKT Tag in media packets is encrypted using a cipher specified via the EKTKey message (e.g., AES Key Wrap with a 128-bit key). This cipher is different than the cipher used to protect media and is only used to encrypt the endpoint's SRTP master key (and other EKT Tag data as per [RFC8870]).

The KEK is not given to the Media Distributor.

6.4. Endpoints Fabricate an SRTP Master Key

As stated earlier, the E2E key determined via DTLS-SRTP **MAY** be discarded in favor of a locally generated E2E SRTP master key. While the DTLS-SRTP-derived SRTP master key can be used initially, the endpoint might choose to change the SRTP master key periodically and **MUST** change the SRTP master key as a result of the EKT key changing.

A locally generated SRTP master key is used along with the master salt transmitted to the endpoint from the Key Distributor via the EKTKey message to encrypt media end to end.

Since the Media Distributor is not involved in E2E functions, it does not create this key, nor does it have access to any endpoint's E2E key. Note, too, that even the Key Distributor is unaware of the locally generated E2E keys used by each endpoint.

The endpoint transmits its E2E key to other endpoints in the conference by periodically including it in SRTP packets in a Full EKT Tag. When placed in the Full EKT Tag, it is encrypted using the EKT Key provided by the Key Distributor. The master salt is not transmitted, though, since all endpoints receive the same master salt via the EKTKey message from the Key Distributor. The recommended frequency with which an endpoint transmits its SRTP master key is specified in [RFC8870].

6.5. Summary of Key Types and Entity Possession

All endpoints have knowledge of the KEK.

Every HBH key is distinct for a given endpoint; thus, Endpoint A and Endpoint B do not have knowledge of the other's HBH key. Since HBH keys are derived from a DTLS-SRTP association, there is at most one HBH key per endpoint. (The only exception is where the DTLS-SRTP association might be rekeyed per Section 5.2 of [RFC5764] and a new key is created to replace the former key.)

Each endpoint generates its own E2E key (SRTP master key); thus, there is a distinct E2E key per endpoint. This key is transmitted (encrypted) via the Full EKT Tag to other endpoints. Endpoints that receive media from a given transmitting endpoint gain knowledge of the transmitter's E2E key via the Full EKT Tag.

Table 2 summarizes the various keys and which entity is in possession of a given key.

Key/Entity	Endpoint A	MD X	MD Y	Endpoint B
KEK (EKT Key)	Yes	No	No	Yes
E2E Key (A and B)	Yes	No	No	Yes
HBH Key (A<=>MD X)	Yes	Yes	No	No
HBH Key (B<=>MD Y)	No	No	Yes	Yes
HBH Key (MD X<=>MD Y)	No	Yes	Yes	No

Table 2: Key Types and Entity Possession

7. Encrypted Media Packet Format

Figure 4 presents a complete picture of what an encrypted media packet per this framework looks like when transmitted over the wire. The packet format shown in the figure is encrypted using the double cryptographic transform with an EKT Tag appended to the end.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<++
      |V=2|P|X|  CC   |M|     PT      |       sequence number         | IO
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ IO
      |                           timestamp                           | IO
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ IO
      |           synchronization source (SSRC) identifier            | IO
      +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ IO
      |            contributing source (CSRC) identifiers             | IO
      |                               ....                            | IO
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+O
      |                    RTP extension (OPTIONAL) ...               | |O
  +>+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+O
  O I |                          payload  ...                         | IO
  O I |                               +-------------------------------+ IO
  O I |                               | RTP padding   | RTP pad count | IO
  O +>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+O
  O | |                    E2E authentication tag                     | |O
  O | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |O
  O | |                            OHB ...                            | |O
  +>| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |+
  | | |                    HBH authentication tag                     | ||
  | | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ||
  | | |   EKT Tag ...   | R                                             ||
  | | +-+-+-+-+-+-+-+-+-+ |                                             ||
  | |                     +- Neither encrypted nor authenticated;       ||
  | |                        appended after the double transform        ||
  | |                        is performed                               ||
  | |                                                                   ||
  | +- E2E-Encrypted Portion               E2E-Authenticated Portion ---+|
  |                                                                      |
  +--- HBH-Encrypted Portion               HBH-Authenticated Portion ----+

      I = Inner (E2E) encryption/authentication
      O = Outer (HBH) encryption/authentication
```

Figure 4: Encrypted Media Packet Format

8. Security Considerations

8.1. Third-Party Attacks

Third-party attacks are attacks attempted by an adversary that is not supposed to have access to keying material or is otherwise not an authorized participant in the conference.

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

Off-path attackers could try connecting to different PERC entities to send specifically crafted packets with an aim of forcing the receiver to forward or render bogus media packets. Endpoints and Media Distributors mitigate such an attack by performing HBH authentication and discarding packets that fail authentication.

Another attack vector is a third party claiming to be a Media Distributor, fooling endpoints into sending packets to the false Media Distributor instead of the correct one. The deceived sending endpoints could incorrectly assume that their packets have been delivered to endpoints when they in fact have not. While this attack is possible, the result is a simple denial of service with no leakage of confidential information, since the false Media Distributor would not have access to either HBH or E2E encryption keys.

A third party could cause a denial of service by transmitting many bogus or replayed packets toward receiving devices and ultimately degrading conference or device performance. Therefore, implementations might wish to devise mechanisms to safeguard against such illegitimate packets, such as utilizing rate-limiting or performing basic sanity checks on packets (e.g., looking at packet length or expected sequence number ranges), before performing decryption operations that are more expensive.

The use of mutual DTLS authentication (as required by DTLS-SRTP) also helps to prevent a denial-of-service attack by preventing a false endpoint or false Media Distributor from successfully participating as a perceived valid media sender that could otherwise carry out an on-path attack. When mutual authentication fails, a receiving endpoint would know that it could safely discard media packets received from the endpoint without inspection.

8.2. Media Distributor Attacks

A malicious or compromised Media Distributor can attack the session in a number of possible ways, as discussed below.

8.2.1. Denial of Service

A simple form of attack is discarding received packets that should be forwarded. This solution framework does not provide any mitigation mechanisms for Media Distributors that fail to forward media packets.

Another form of attack is modifying received packets before forwarding. With this solution framework, any modification of the E2E-authenticated data results 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 along with a Full EKT Tag. Since such a message would trigger the receiver to form a new cryptographic context, the Media Distributor can attempt to consume the receiving endpoint's resources. While E2E authentication would fail and the cryptographic context would be destroyed, the key derivation operation would nonetheless consume some computational resources. While resource consumption attacks cannot be mitigated entirely, rate-limiting packets might help reduce the impact of such attacks.

8.2.2. Replay Attacks

A replay attack is an attack where an already-received packet from a previous point in the RTP stream is replayed as a 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.

A replay attack is mitigated by the requirement to implement replay protection as described in Section 3.3.2 of [RFC3711]. E2E replay protection **MUST** be provided for the duration of the conference.

8.2.3. Delayed Playout Attacks

A delayed playout attack is an attack where media is received and held by a Media Distributor and then forwarded to endpoints at a later point in time.

This attack is possible even if E2E replay protection is in place. Because the Media Distributor is allowed to select a subset 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 problem here is that a Media Distributor can choose to not deliver a particular stream for a while.

While the Media Distributor can purposely stop forwarding media flows, it can also select an arbitrary starting point to resume forwarding those media flows, perhaps forwarding old packets rather than current packets. As a consequence, what the media source sent can be substantially delayed at the receiver with the receiver believing that newly arriving packets are delayed only by transport delay when the packets may actually be minutes or hours old.

While this attack cannot be eliminated entirely, its effectiveness can be reduced by rekeying the conference periodically, since significantly delayed media encrypted with expired keys would not be decrypted by endpoints.

8.2.4. Splicing Attacks

A splicing attack is an attack where a Media Distributor receiving multiple media sources splices one media stream into the other. If the Media Distributor were 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 that stream A was previously using. By including the SSRC in the integrity check for each packet -- both HBH and E2E -- PERC prevents splicing attacks.

8.2.5. RTCP Attacks

PERC does not provide E2E protection of RTCP messages. This allows a compromised Media Distributor to impact any message that might be transmitted via RTCP, including media statistics, picture requests, or loss indication. It is also possible for a compromised Media Distributor to forge requests, such as requests to the endpoint to send a new picture. Such requests can consume significant bandwidth and impair conference performance.

8.3. Key Distributor Attacks

As stated in Section 3.2.2, the Key Distributor needs to be secured, since exploiting the Key Server can allow an adversary to gain access to the keying material for one or more conferences. Having access to that keying material would then allow the adversary to decrypt media sent from any endpoint in the conference.

As a first line of defense, the Key Distributor authenticates every security association -- associations with both endpoints and Media Distributors. The Key Distributor knows which entities are authorized to have access to which keys, and inspection of certificates will substantially reduce the risk of providing keys to an adversary.

Both physical and network access to the Key Distributor should be severely restricted. This may be more difficult to achieve when the Key Distributor is embedded within, for example, an endpoint. Nonetheless, consideration should be given to shielding the Key Distributor from unauthorized access or any access that is not strictly necessary for the support of an ongoing conference.

Consideration should be given to whether access to the keying material will be needed beyond the conclusion of a conference. If not needed, the Key Distributor's policy should be to destroy the keying material once the conference concludes or when keying material changes during the course of the conference. If keying material is needed beyond the lifetime of the conference, further consideration should be given to protecting keying material from future exposure. While it might seem obvious, it is worth making this point, to avoid any doubt that if an adversary were to record the media packets transmitted during a conference and then gain unauthorized access to the keying material left unsecured on the Key Distributor even years later, the adversary could decrypt the content of every packet transmitted during the conference.

8.4. Endpoint Attacks

A Trusted Endpoint is so named because conference confidentiality relies heavily on the security and integrity of the endpoint. If an adversary successfully exploits a vulnerability in an endpoint, it might be possible for the adversary to obtain all of the keying material used in the conference. With that keying material, an adversary could decrypt any of the media flows received from any other endpoint, either in real time or at a later point in time (assuming that the adversary makes a copy of the media packets).

Additionally, if an adversary successfully exploits an endpoint, the adversary could inject media into the conference. For example, an adversary could manipulate the RTP or SRTP software to transmit whatever media the adversary wishes to send. This could involve the reuse of the compromised endpoint's SSRC or, since all conference participants share the same KEK, the use of a new SSRC or the SSRC value of another endpoint. Only a single SRTP cipher suite defined provides source authentication properties that allow an endpoint to cryptographically assert that it sent a particular E2E-protected packet (namely, Timed Efficient Stream Loss-Tolerant

Authentication (TESLA) [RFC4383]), and its usage is presently not defined for PERC. The suite defined in PERC only allows an endpoint to determine that whoever sent a packet had received the KEK.

However, attacks on the endpoint are not limited to the PERC-specific software within the endpoint. An attacker could inject media or record media by manipulating the software that sits between the PERC-enabled application and the hardware microphone of a video camera, for example. Likewise, an attacker could potentially access confidential media by accessing memory, cache, disk storage, etc. if the endpoint is not secured.

9. IANA Considerations

This document has no IANA actions.

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>.

[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>.

[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>.

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

[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>.

[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>.

[RFC8870] Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", RFC 8870, DOI 10.17487/RFC8870, October 2020, <https://www.rfc-editor.org/info/rfc8870>.

10.2. Informative References

[PERC-DTLS] Jones, P. E., Ellenbogen, P. M., and N. H. Ohlmeier, "DTLS Tunnel between a Media Distributor and Key Distributor to Facilitate Key Exchange", Work in Progress, Internet-Draft, draft-ietf-perc-dtls-tunnel-06, 16 October 2019, <https://tools.ietf.org/html/draft-ietf-perc-dtls-tunnel-06>.

[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, <https://www.rfc-editor.org/info/rfc3261>.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, DOI 10.17487/RFC4353, February 2006, <https://www.rfc-editor.org/info/rfc4353>.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <https://www.rfc-editor.org/info/rfc4383>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://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, <https://www.rfc-editor.org/info/rfc5763>.

[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>.

[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, <https://www.rfc-editor.org/info/rfc6464>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <https://www.rfc-editor.org/info/rfc7667>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, October 2020, <https://www.rfc-editor.org/info/rfc8827>.

[W3C.CR-webrtc-20191213] Bergkvist, A., Burnett, D., Jennings, C., Narayanan, A., Aboba, B., Brandstetter, T., Boström, H., and J-I. Bruaroey, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium CR CR-webrtc-20191213, December 2019, <https://www.w3.org/TR/webrtc/>.

Acknowledgments

The authors would like to thank Mo Zanaty, Christian Oien, and Richard Barnes for invaluable input on this document. Also, we would like to acknowledge Nermeen Ismail for serving on the initial draft versions of this document as a coauthor. We would also like to acknowledge John Mattsson, Mats Naslund, and Magnus Westerlund for providing some of the text in the document, including much of the original text in the Security Considerations section (Section 8).

Authors' Addresses

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

David Benham
Independent
California
United States of America
Email: dabenham@gmail.com

Christian Groves
Independent
Melbourne
Australia
Email: cngroves.std@gmail.com