

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 8, 2019

P. Jones
Cisco
D. Benham
C. Groves
Independent
September 4, 2018

**A Solution Framework for Private Media in Privacy Enhanced RTP
Conferencing
draft-ietf-perc-private-media-framework-07**

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 aims to build upon existing security mechanisms defined for the real-time transport protocol (RTP).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 8, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Conventions Used in This Document	4
3.	PERC Entities and Trust Model	5
3.1.	Untrusted Entities	5
3.1.1.	Media Distributor	6
3.1.2.	Call Processing	6
3.2.	Trusted Entities	7
3.2.1.	Endpoint	7
3.2.2.	Key Distributor	7
4.	Framework for PERC	7
4.1.	End-to-End and Hop-by-Hop Authenticated Encryption	8
4.2.	E2E Key Confidentiality	9
4.3.	E2E Keys and Endpoint Operations	9
4.4.	HBH Keys and Hop Operations	10
4.5.	Key Exchange	10
4.5.1.	Initial Key Exchange and Key Distributor	11
4.5.2.	Key Exchange during a Conference	12
5.	Authentication	13
5.1.	Identity Assertions	13
5.2.	Certificate Fingerprints in Session Signaling	13
5.3.	Conferences Identification	14
6.	Security Considerations	14
6.1.	Third Party Attacks	14
6.2.	Media Distributor Attacks	15
6.2.1.	Denial of service	15
6.2.2.	Replay Attack	16
6.2.3.	Delayed Playout Attack	16
6.2.4.	Splicing Attack	16
7.	IANA Considerations	16
8.	Acknowledgments	17
9.	References	17
9.1.	Normative References	17
9.2.	Informative References	17
Appendix A.	PERC Key Inventory	19
A.1.	DTLS-SRTP Exchange Yields HBH Keys	20
A.2.	The Key Distributor Transmits the KEK (EKT Key)	20
A.3.	Endpoints fabricate an SRTP Master Key	21
A.4.	Who has What Key	21
Appendix B.	PERC Packet Format	22
	Authors' Addresses	23

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 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 RTCP packets. 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.

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 terms and acronyms:

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: 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 have access to E2E encryption keys. It operates according to the Selective Forwarding Middlebox RTP topologies [\[RFC7667\]](#) 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 distributes keying material and related information to trusted endpoints and Media Distributor(s), only that which 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.

Call Processing: All trusted endpoints in the conference connect to it by a call processing dialog, such as with the Focus defined in the Framework for Conferencing with 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

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.

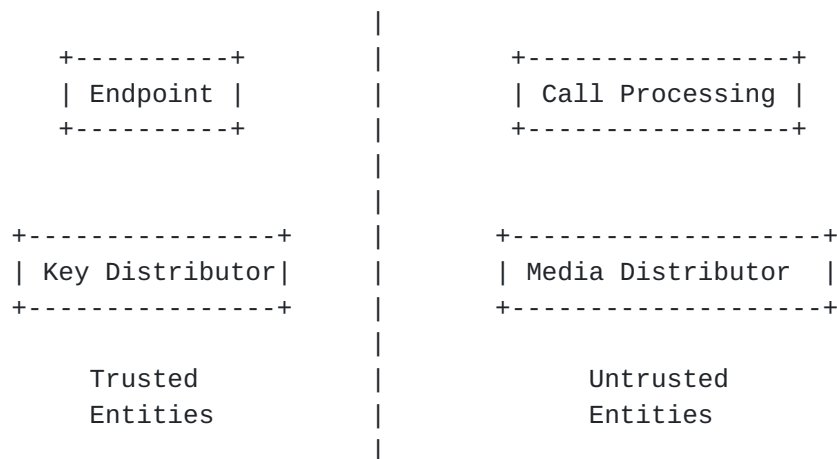


Figure 1: Trusted and Untrusted Entities in PERC

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.

It is important to understand that untrusted in this document does not mean 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 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** be decryptable by a Media Distributor, so it is untrusted to have access to the E2E media encryption keys. The key exchange mechanisms specified in this framework will prevent the Media Distributor from gaining access to the E2E media encryption keys.

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. Instead, the Key Distributor is responsible for authenticating endpoints (e.g., using WebRTC Identity [[I-D.ietf-rtcweb-security-arch](#)]) and determining their authorization to receive E2E 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 6](#)), etc, to a level equal to or better than traditional conferencing (i.e. non-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 keys 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., non-PERC) deployments.

[3.2.2.](#) Key Distributor

The Key Distributor, which may be colocated 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 is necessary to 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 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](#)].

for a description of the keys used in PERC and [Appendix B](#) for an overview of how the packet appears on the wire.

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 the SRTP master key used for E2E authenticated encryption of media sent by a given endpoint. Each endpoint in the conference will create a random SRTP master key for E2E authenticated encryption, thus participants in the conference MUST keep track of the E2E keys received via the Full EKT Field for each distinct SSRC in the conference so that it can properly decrypt received media. Note, too, that an endpoint may change its E2E key at any time and advertise that new key to the conference as specified in [\[I-D.ietf-perc-srtp-ekt-diet\]](#).

[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 information. Following a change in an E2E key, prior E2E keys SHOULD be retained by receivers for a period long enough to ensure that late-arriving or out-of-order packets from the endpoint can be successfully decrypted. Receiving endpoints MUST discard old E2E keys 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 an E2E key.

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. Each HBH key is distinct per hop and no two hops ever 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 HBH, 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

In brief, the keys used by any given endpoints are determined in the following way:

- o 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).
- o The E2E key that an endpoint uses to send SRTP media can either be set from DTLS or chosen by the endpoint. It is then distributed to other endpoints in a Full EKT Field, encrypted under an EKTKey provided to the client by the Key Distributor within the DTLS channel they negotiated.
- o Each E2E key that an endpoint uses to receive SRTP media is set by receiving a Full EKT Field from another endpoint.

4.5.1. Initial Key Exchange and Key Distributor

The Media Distributor maintains a tunnel with the Key Distributor (e.g., using [[I-D.ietf-perc-dtls-tunnel](#)]), 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 the following figure. The DTLS association between endpoints and the Key Distributor will enable each endpoint to generate E2E and HBH keys and receive the Key Encryption Key (KEK) (i.e., EKT Key). 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.

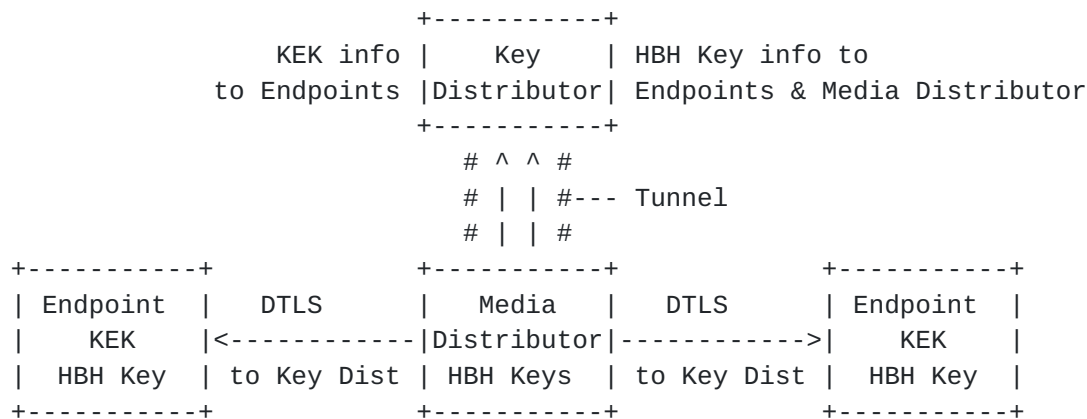


Figure 3: Exchanging Key Information Between Entities

Endpoints will establish a DTLS-SRTP [[RFC5764](#)] 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 is 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.

In 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 Key Encryption Key (KEK) (i.e., EKT Key) is conveyed by the Key Distributor over the DTLS association to endpoints via procedures defined in PERC EKT [[I-D.ietf-perc-srtp-ekt-diet](#)] via the EKTKey message.

Note that following DTLS-SRTP procedures for the [\[I-D.ietf-perc-double\]](#) cipher, the endpoint will generate both E2E and HBH encryption keys and salt values. Endpoints MAY use the DTLS-SRTP generated E2E key for transmission or MAY 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 [\[I-D.ietf-perc-srtp-ekt-diet\]](#).

Media Distributors use DTLS-SRTP [\[RFC5764\]](#) 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 will be able to encrypt media end-to-end with an E2E key, sending that E2E key to other endpoints encrypted with the KEK, and will be able to encrypt and authenticate RTP packets using a HBH key. 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 re-key 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 EKTKey. 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 [\[I-D.ietf-perc-srtp-ekt-diet\]](#).

Endpoints MAY follow the procedures in [section 5.2 of \[RFC5764\]](#) to re-negotiate 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 [[I-D.ietf-perc-dtls-tunnel](#)].

Endpoints are at liberty to change the E2E encryption key used at any time. Endpoints MUST generate a new E2E encryption key whenever it receives a new EKT Key. After switching to a new key, the new key will be conveyed to other endpoints in the conference in RTP/RTCP packets per [[I-D.ietf-perc-srtp-ekt-diet](#)].

5. Authentication

It is important to this solution framework that the entities can 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 [[I-D.ietf-rtcweb-security-arch](#)] 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 to bind 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. 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 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 by the Focus [\[RFC4353\]](#). 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.

[5.3.](#) Conferences Identification

The Key Distributor needs to know what endpoints are being added to a given conference. Thus, the Key Distributor and the Media Distributor will need to know endpoint-to-conference mappings, which is enabled by exchanging a conference-specific unique identifier as defined in [\[I-D.ietf-perc-dtls-tunnel\]](#). How this unique identifier is assigned is outside the scope of this document.

[6.](#) Security Considerations

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. IANA Considerations

There are no IANA considerations for this document.

8. 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.

9. References

9.1. Normative References

- [I-D.ietf-perc-double]
Jennings, C., Jones, P., Barnes, R., and A. Roach, "SRTP Double Encryption Procedures", [draft-ietf-perc-double-09](#) (work in progress), May 2018.
- [I-D.ietf-perc-dtls-tunnel]
Jones, P., Ellenbogen, P., and N. Ohlmeier, "DTLS Tunnel between a Media Distributor and Key Distributor to Facilitate Key Exchange", [draft-ietf-perc-dtls-tunnel-03](#) (work in progress), April 2018.
- [I-D.ietf-perc-srtp-ekt-diet]
Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", [draft-ietf-perc-srtp-ekt-diet-08](#) (work in progress), July 2018.
- [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>>.
- [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>>.

9.2. Informative References

- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", [draft-ietf-rtcweb-security-arch-15](#) (work in progress), July 2018.

- [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>>.
- [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>>.
- [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>>.
- [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, <<https://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, <<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., Iovov, 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>>.

Appendix A. PERC Key Inventory

PERC specifies the use of a number of different keys and, understandably, it looks complicated or confusing on the surface. This section summarizes the various keys used in the system, 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 Figure 4 below.

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

Figure 4: Key Inventory

As you can see, the number key types is very small. However, what can be challenging is keeping track of all of the distinct E2E keys as the conference grows in size. With 1,000 participants in a conference, there will be 1,000 distinct SRTP master keys, all of which share the same master salt. Each of those keys are passed through the KDF 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. And, of course, there is a new SRTP master salt to go with those keys. Endpoints have to retain old keys for a period of time to ensure they can properly decrypt late-arriving or out-of-order packets.

The time required to retain old keys (either EKT Keys or SRTP master keys) is not specified, but they should be retained at least for the period of time required to re-key the conference or handle late-arriving or out-of-order packets. A period of 60s should be considered a generous retention period, but endpoints may keep old keys on hand until the end of the conference.

Or more detailed explanation of each of the keys follows.

A.1. DTLS-SRTP Exchange Yields HBH Keys

The first set of keys acquired are for hop-by-hop encryption and decryption. Assuming the use of Double [[I-D.ietf-perc-double](#)], the endpoint would perform DTLS-SRTP exchange with the key distributor and receive a key that is, in fact, "double" the size that is needed. Per the Double specification, the E2E part is the first half of the key, so the endpoint will just discard that information in PERC. It is not used. The second half of the key 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 media distributor doesn't perform DTLS-SRTP, but it is at this point that the key distributor will inform the media distributor of the HBH key value via the tunnel protocol ([\[I-D.ietf-perc-dtls-tunnel\]](#)). The key distributor will send the least significant bits corresponding to the half of the keying material determined through DTLS-SRTP with the endpoint to the media distributor via the tunnel protocol. There is a salt 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 (i.e., half of the generated salt material) are sent to the media distributor via the tunnel protocol.

No two endpoints will have the same HBH key, thus the media distributor must keep track each distinct HBH key (and the corresponding salt) and use it only for the specified hop.

This key is also used for HBH encryption of RTCP. RTCP is not end-to-end encrypted in PERC.

A.2. The Key Distributor Transmits the KEK (EKT Key)

Via the aforementioned DTLS-SRTP association, the key distributor will send the endpoint the KEK (i.e., EKT Key per [\[I-D.ietf-perc-srtp-ekt-diet\]](#)). This key is known only to the key distributor and endpoints. This key is the most important to protect since having knowledge of this key (and the SRTP master salt transmitted as a part of the same message) will allow 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 can be used to re-key the entire conference. Each key is identified by a "Security Parameter Index" (SPI) value. Endpoints should expect that a conference might be re-keyed 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 EKT Field in media packets is encrypted using a cipher specified via the EKKey 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 Field data as per [[I-D.ietf-perc-srtp-ekt-diet](#)]).

The media distributor is not given the KEK (i.e., EKT Key).

[A.3.](#) 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 SRTP master key. While the DTLS-SRTP-derived key could be used, the fact that an endpoint might need to change the SRTP master key periodically or is forced to change the SRTP master key as a result of the EKT key changing means using it has only limited utility. To reduce complexity, PERC *RECOMMENDS* that endpoints create random SRTP master keys locally to be used for E2E encryption.

This locally-generated SRTP master key is used along with the master salt transmitted to the endpoint from the key distributor via the EKKey message to encrypt media end-to-end.

Since the media distributor is not involved in E2E functions, it will not create this key nor 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 will transmit its E2E key to other endpoints in the conference by periodically including it in SRTP packets in a Full EKT Field. When placed in the Full EKT Field, it is encrypted using the EKT Key provided by the key distributor. The master salt is not transmitted, though, since all endpoints will have received the same master salt via the EKKey message. The recommended frequency with which an endpoint transmits its SRTP master key is specified in [[I-D.ietf-perc-srtp-ekt-diet](#)].

[A.4.](#) Who has What Key

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.

Each endpoint generates its own E2E Key (SRTP master key), thus the key distinct per endpoint. This key is transmitted (encrypted) via the EKT Field to other endpoints. Endpoints that receive media from a given transmitting endpoint will therefore have knowledge of the transmitter's E2E key.

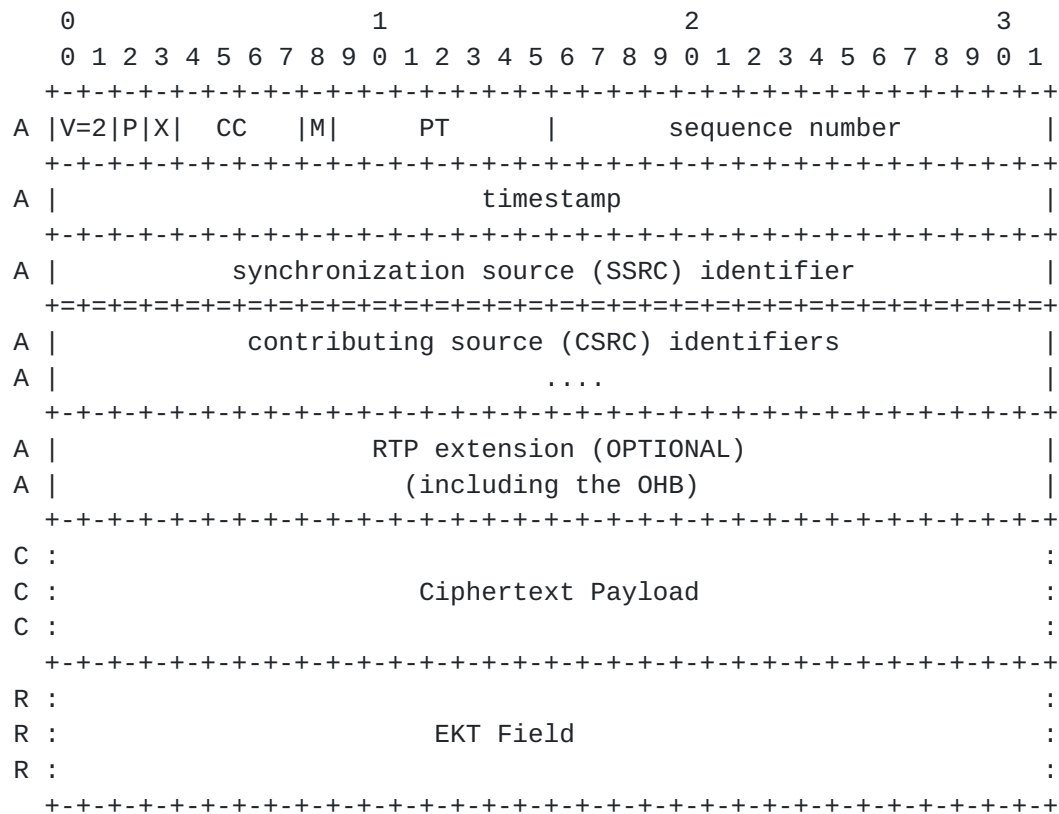
To summarize the various keys and which entity is in possession of a given key, refer to Figure 5.

Key / Entity	Endpoint A	MD X	MD Y	Endpoint B
KEK	Yes	No	No	Yes
E2E Key (A and B)	Yes	No	No	Yes
HBH Key (A \leftrightarrow MD X)	Yes	Yes	No	No
HBH Key (B \leftrightarrow MD Y)	No	No	Yes	Yes
HBH Key (MD X \leftrightarrow MD Y)	No	Yes	Yes	No

Figure 5: Keys per Entity

[Appendix B](#). PERC Packet Format

Figure 6 presents a complete picture of what a PERC packet looks like when transmitted over the wire.



C = Ciphertext (encrypted and authenticated)
A = Associated Data (authenticated only)
R = neither encrypted nor authenticated, added
after Authenticated Encryption completed

Figure 6: PERC Packet Format

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
Independent

Email: dabenham@gmail.com

Christian Groves
Independent
Melbourne
Australia

Email: Christian.Groves@nteczone.com