Network Working Group
Internet Draft
Intended status: Standards Track
Expires: January 6, 2016

P. Jones (Ed.)
N. Ismail
D. Benham
Cisco Systems

July 6, 2015

A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing
draft-jones-perc-private-media-framework-00

Abstract

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

Table of Contents

1. Introduction

Switched conferencing is an increasingly popular model for multimedia conferences with multiple participants using a combination of audio, video, text, and other media types. With this model, real-time media flows from conference participants are not mixed, transcoded, transrated, recomposed, or otherwise manipulated by a media distribution device (MDD), 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 MDD 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 switching MDDs 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 MDDs can be deployed on general-purpose computing hardware. This, in turn, means that it is possible to deploy switching MDDs in virtualized environments, including private and public clouds. Deploying conference resource in a 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 the hardware. Additionally, there are usually several ports open to the public in cloud deployments, such as for remote administration, and so on.

Recognizing the need to improve the way in which media confidentiality is ensured, requirements for private media were specified in [I.D-draft-jones-perc-private-media-reqts]. Attempting to meet those requirements, this document defines a solution framework wherein privacy is ensured by making it impossible for an MDD 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 switching MDD 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 switching MDD by using a key that is independent from the media encryption and authentication key(s) and is unique to the participating endpoint and the switching MDD.

A goal of this framework is to meet the referenced requirements and stated objectives by utilizing existing security procedures defined for RTP with minimal extensions.

2. Requirements Language

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 RFC 2119 [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.

3. Private Media Trust Model

The private media trust model is specified in [I.D-draft-jones-perc-private-media-reqts].

4. Solution Framework Overview

The purpose for this framework is to define a means through which media privacy can be ensured when communicating within a switched conferencing environment consisting of one or more centrally located media distribution devices. This framework specifies the reuse of

several technologies, including SRTP [RFC3711], EKT [I.D-draft-ietf-avtcore-srtp-ekt], and DTLS-SRTP [RFC5764]. For the purposes of this document, a conference refers to any session with two or more participants - endpoints or other trusted devices - exchanging RTP flows through media distribution devices.

4.1. End-to-End Media Privacy

This framework does not attempt to hide the fact that communication between parties takes place. Rather, it only addresses the end-to-end confidentiality and integrity of the actual media content.

To ensure the confidentiality and integrity of RTP media packets, endpoints will utilize an EKT key - known to all conference participants - to encrypt the SRTP key that is used to encrypt the media (i.e., the RTP payload) via authenticated encryption.

Note that this 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 when a conference is to be re-keyed is outside the scope of this document, but this framework does enable re-keying of the conference.

Endpoints MUST maintain a list of SSRCs, track received sequence number values relating to those SSRCs, and maintain associated SRTP master keys for those SSRCs. All of this information SHOULD be retained for some reasonable period of time and SHOULD be discarded shortly after the EKT key for the conference is changed and upon leaving the conference. However, following a change of the EKT key, old key material SHOULD be retained long enough to ensure that late-arriving or out-of-order packets can be successfully played.

4.2. Hop-by-Hop Operations

To ensure the integrity of transmitted media packets, this framework requires that every packet be authenticated hop-by-hop. The authentication key used for hop-by-hop authentication is derived from an SRTP master key shared only on the respective hop between the endpoint and the MDD to which it is attached. If MDDs are cascaded, then there will also be an SRTP master key and derived authentication key shared between the cascaded servers. Importantly, each of these keys is distinct per hop and no two hops ever intentionally use the same SRTP master key.

MDDs may find it necessary to change certain parts of the RTP packet header, add or remove RTP header extensions, etc. By using hop-by-hop authentication, the MDD is given liberty to change certain values present in the RTP header, such as the payload type value.

If there is a desire to encrypt RTP header extensions, an encryption key is derived from the hop-by-hop SRTP master key to encrypt header

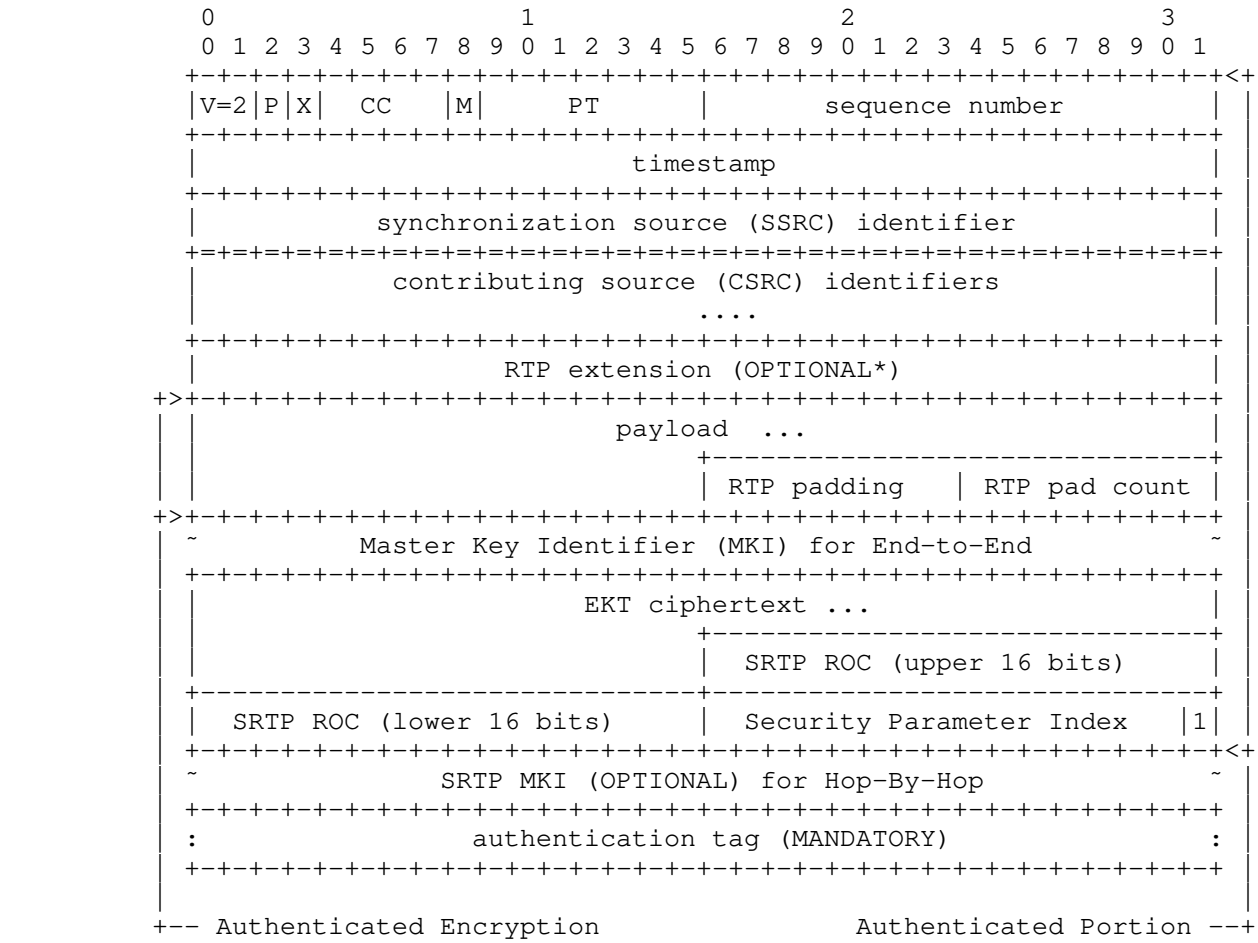
extensions as per [RFC6904]. This will give the switching MDD visibility into header extensions, such as the one used to determine audio level [RFC6464] of conference participants. Note that allowing RTP header extensions to be encrypted requires that all hops decrypt and re-encrypt any encrypted header extensions.

RTCP is optionally encrypted and mandatorily authenticated hop-by-hop using the encryption and authentication keys derived from the SRTP master key for the hop. This gives the switching MDD the flexibility of either forwarding RTCP packets unchanged, transmit compound RTCP packets, or to create RTCP packets to report statistics or for conference control.

One of the reasons for performing hop-by-hop authentication is to provide replay protection. If a media packet is replayed to the switching MDD, it will be detected and rejected. Likewise, the endpoint can detect replayed packets originally sent by the MDD. Packets received by an endpoint that were originally sent to a different endpoint will fail to pass authentication checks.

5. Private Media Packet Format

Since the RTP packet payload is encrypted and authenticated end-to-end, extensions optionally encrypted hop-by-hop, and the entire RTP packet is authenticated hop-by-hop, it may be useful to see the entire RTP packet similarly to what is shown in [RFC3711].

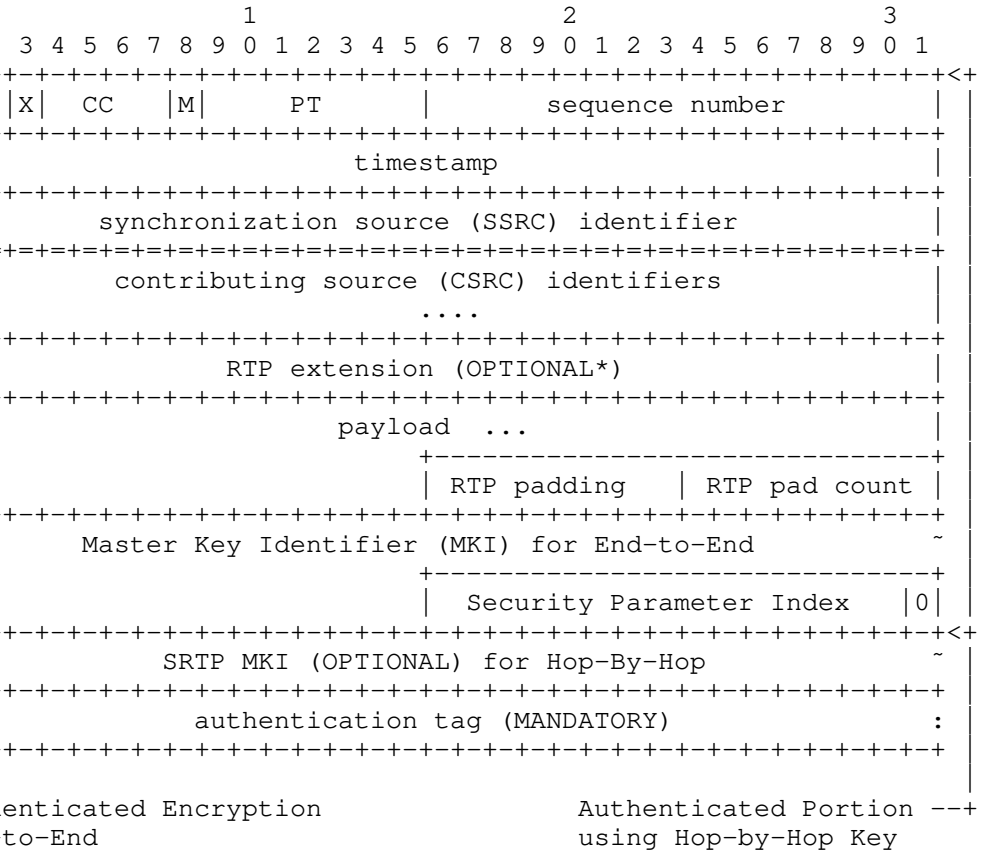


* Header extensions are optionally Encrypted Hop-by-Hop

Figure 1 - Private Media SRTP Packet with Full EKT Field

The rollover counter value is shown and transmitted as plaintext. This is necessary since a switching MDD may not transmit media from a "silent" participant's endpoint to others in the conference for a long period of time. When media from that previously "silent" participant is later forwarded to others, the receiving endpoint(s) would not otherwise know the value of the rollover counter. Further, this value is needed so that the correct authentication tag can be generated hop-by-hop. Since the expected length of the EKT Field might not be known to the MDD that is only authenticating packets, the ROC field is placed as shown to ensure that its location can be consistently determined.

The EKT Field shown in Figure 1 is the "Full EKT Field". The "Short EKT Field" may also be present in its place. When the short EKT Field is used, the packet format looks like the one shown in Figure 2.



* Header extensions are optionally Encrypted Hop-by-Hop

Figure 2 - Private Media SRTP Packet with Short EKT Field

Note that the ROC field is absent when the short EKT Field is transmitted. The assumption is that endpoints will transmit the Full EKT Field regularly and frequently (e.g., every 100ms) to ensure that media transmitted by a previously "silent" participant's endpoint can be properly decrypted by other endpoints within a period of time that is not noticeable to the human user.

6. SRTP Cryptographic Context

For any given media source identified by its SSRC, there is a single SRTP cryptographic context as described in Section 3.2 of [RFC3711] used in this framework.

For end-to-end encryption, this framework extends the parameter set of the cryptographic context by adding an identifier for the end-to-end authenticated encryption algorithm. That parameter has associated with it an EKT key (and associated EKT information, such as master salt, key length, etc.), one or more SRTP master keys, and as outlined in Section 3.2.1 of [RFC3711], other associated values that relate to the master keys (e.g., master salt and key length values).

For hop-by-hop encryption, the existing parameters in the SRTP cryptographic context are used, including for the optional encryption of RTP header extensions, authentication tag generation, etc.

7. Cryptographic Operations

7.1. Hop-by-Hop Authentication and Optional Encryption

For operations that occur hop-by-hop, the cryptographic transforms defined in SRTP [RFC3711] (or other standardized transforms) may be used in order optionally encrypt RTP header extensions, authenticate the RTP packet, optionally encrypt the RTCP packet, and to authenticate the RTCP packet.

The encryption and authentication of the RTP payload (media content) itself is not a hop-by-hop operation, as explained in the next section.

The procedures for optionally encrypting RTP header extensions is define in [RFC6904] and MUST be used when encrypting header extensions using the hop-by-hop SRTP master key to derive the k_he and k_hs values.

The procedures for authenticating the RTP packet, optionally encrypting the RTCP packet, and for authenticating the RTCP packet shall follow the procedures defined in [RFC3711] using the hop-by-hop SRTP master key and master salt to derive additional keys as specified in that specification.

7.2. End-to-End Media Payload Encryption and Authentication

This section covers the encryption and authentication of the RTP payload (i.e., media content) using the SRTP master key(s) created by the endpoint and securely conveyed in the EKT Field using the EKT key(s) shared with each of the endpoints in the conference.

This framework requires that the end-to-end cryptographic transforms use authenticated encryption with associated data (AEAD) algorithms. Specifically, the transforms defined in [I.D-draft-ietf-avtcore-srtp-aes-gcm] are used as the mandatory transforms in this framework.

The procedures followed to encrypt the payload are those described in [I.D-draft-ietf-avtcore-srtp-aes-gcm], except that the associated data used with those algorithms specified in Section 8.2 is redefined as follows:

Associated Data: The version V (2 bits), padding flag P (1 bit), CSRC count CC (4 bits), marker M (1 bit), the sequence number (16 bits), timestamp (32 bits), SSRC (32 bits), and optional contributing source identifiers (CSRCs, 32 bits each).

The authentication tag for the end-to-end encrypted payload immediately follows the encrypted payload in the defined packet format (see section 5).

Note that RTP header extensions are not encrypted as a part of the end-to-end operations. Rather, they are encrypted as a hop-by-hop operation as explained in the previous section.

Only a part of the RTP packet is authenticated with the above definition of "Associated Data" since packets are authenticated hop-by-hop and there is a desire to allow switching MDDs to make changes to certain parts of the RTP header. For these reasons, there is a need for an authentication tag as defined in [RFC3711] to be placed at the end of the RTP packet. This authentication tag is provided via the hop-by-hop authentication operation as discussed in the previous section. Note that this is also a deviation from what [I.D-draft-ietf-avtcore-srtp-aes-gcm][DB1] recommends, but is necessary to allow the switching MDD to make changes to certain fields that would otherwise be protected.

7.2.1. End-to-End Cryptographic Context Considerations

7.2.1.1. Initialization Vector Formation

SRTP defines the following Initialization Vector (IV) as part of the context for the AES Counter Mode cipher when encrypting RTP packets:

```
SRTP IV =
    (SALT << 16) XOR (SSRC << 64) XOR (ROC << 32) XOR (SEQ << 16)
```

Following similar logic, [I.D-draft-ietf-avtcore-srtp-aes-gcm] defines an Initialization Vector for encrypting RTP as follows:

```
SRTP IV =
    SALT XOR (0x00 || 0x00 || SSRC ||ROC || SEQ)
```

Since this context includes and makes use of the SSRC, SEQ, and ROC, these parameters must be preserved end-to-end for proper cipher operation. In some RTP topologies, for example, a Selective Forwarding Middlebox (SFM) with a common SSRC space, these parameters

are preserved end-to-end because RTP middleboxes do not alter them. In other RTP topologies, such as a Media Switching Mixer or a SFM with separate SSRC spaces, the RTP middleboxes may need to alter them for proper operation. When any of these parameters is altered, the original parameters must still be preserved elsewhere in the packet since they are essential parts of the cipher context. An RTP header extension for End-to-End IV (EEIV) is defined for forwarding this original context.

7.2.1.2. End-to-End IV (EEIV) RTP Header Extension

The End-to-End IV (EEIV) RTP Header Extension [RFC5285] conveys some or all of the original end-to-end cipher context parameters: SSRC, SEQ, and ROC. The extension has the following format:

EEIV = SSRC || SEQ

The extension MUST be added (if absent) by any RTP middlebox that alters these parameters in the RTP header. It MUST NOT be added, removed or altered if already present. Endpoints MAY add this extension to operate with a RTP middlebox that can forward the extension, but not add such an extension itself.

When an endpoint receives this extension in an SRTP packet, the endpoint MUST use these values as the SSRC and RTP sequence number when performing the authenticated decryption step as opposed to the values found in the RTP header.

Note that he ROC does not need to be present in the header extension since it is included in the payload as proposed in Section 5.

8. Key Exchange

Within this framework, there are various keys each endpoint needs: those for end-to-end encryption/authentication and those for hop-by-hop authentication, optional encryption of RTP header extensions, SRTCP authentication, and optional SRTCP encryption. Likewise, the MDD needs a hop-by-hop key when communicating with an endpoint or cascaded conference server. The challenge is in securely exchanging these keys between the appropriate entities.

To facilitate key exchange, this framework utilizes DTLS-SRTP and procedures defined in EKT. This is explained further in the following sub-sections.

8.1. Session Signaling

The session signaling protocol is not significant to this specification, since the call processing functions are not assumed to be trusted. Signaling might be via SIP [RFC3261] or a proprietary signaling between a browser and a server, as examples. What is

important is that the signaling convey, in some manner, the fingerprint of the endpoint's certificate that will be used with DTLS-SRTP. For the sake of providing a more concrete discussion, it is assumed that SIP is used and SDP [RFC4566] conveys the fingerprint information per [RFC5763].

The endpoint ("User Agent" in SIP terminology) will send an INVITE message containing SDP for the media session along with the endpoint's certificate fingerprint. This message or part thereof MUST be cryptographically signed so as to prevent unauthorized, undetectable modification of the fingerprint value, or the message MUST be sent to a trusted element over a secure connection.

For this example, it is assumed that the endpoint sends a message to a call processing function (e.g., a B2BUA) over a TLS connection. The B2BUA might sign the message using the procedures described in [RFC4474] for the benefit of forwarding the message to other entities. It's important to note, however, that this does not lend to the security of media, as the call processing function is not assumed to be trusted.

An associated Key Management Function (KMF) needs to receive information about the call and the endpoint(s). This might be performed via an interface between the endpoint and the KMF, the call processing function and the KMF, or it might be via a signaling interface between the MDD and the KMF (see Figure 6 in [I.D-draft-jones-perc-private-media-reqts]).

Regardless of the exact method, it is important that the endpoint's certificate fingerprint and a participant identifier (a random value created by the endpoint and provided to the KMF for each RTP session) are securely conveyed to the KMF. The client certificate and participant identifier will allow the KMF to associate the DTLS connection to the specific endpoint and RTP session for the conference.

Ultimately, a call is established to a conference and the endpoint receives address information to which it may establish one or more RTP sessions through to a MDD.

Call signaling going back to the endpoint might contain the certificate fingerprint of the KMF that will process DTLS-SRTP messages. Alternatively, the endpoint might already know the certificate fingerprint. Whatever mechanism is employed, it is extremely vital that the endpoint be able to fully trust the validity of the fingerprint information for the KMF.

8.2. Negotiating SRTP Protection Profiles and Key Exchange

8.2.1. Endpoint and KMF

There is a need for an SRTP master key and STRP master salt for hop-by-hop authentication and optional encryption known to the endpoint and the MDD. Additionally, there is a need to exchange an EKT master key and EKT master salt for the end-to-end encryption of the media content that is known to all participants in the conference, but not known to the switching MDD.

To convey keys, the endpoint uses the procedures defined in [I.D-draft-ietf-avtcore-srtp-ekt] for DTLS-SRTP over the media ports for the RTP session. However, the switching MDD does not terminate the DTLS signaling. Rather, DTLS packets received by the switching MDD are forwarded to the KMF and vice versa. The figure below depicts this.

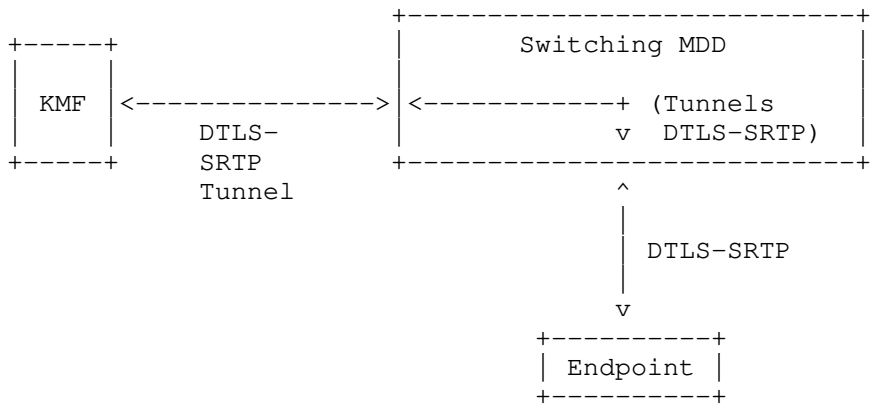


Figure 3 - DTLS-SRTP Tunneled to KMF

Through this tunneled DTLS-SRTP exchange, an EKT master key and EKT master salt are conveyed from the KMF to the endpoint, which the endpoint will use when conveying SRTP keys and encrypt and authenticate the media content in SRTP packets. The DTLS-SRTP message exchanges between the switching MDD and KMF are encapsulated in a second DTLS connection wherein the KMF also provides the MDD with the hop-by-hop key material.

The KMF is described as a logical function in this document where the functionality needed might be provided by one or more physical or virtual entities. For example, there would obviously be a device needed to terminate the DTLS-SRTP signaling. However, that device may or may not be in possession of the EKT key used for the conference. There DTLS-SRTP termination function might interface with a Key Management Server (KMS), such as the one described in [I.D-draft-abiggs-saag-key-management-service].

{Editor's Note: a DTLS encapsulation protocol has been selected, but has not been published in a separate draft. If there is no objection to this approach, a proposal draft for the tunneling protocol will be prepared.}

The endpoint does not transmit media encryption keys to the KMF. The endpoint will follow the procedures specified in the EKT specification to generate an SRTP master key and convey this information to conference participants periodically (and anytime an I-Frame is explicitly requested) via the "Full EKT Field."

This tunneling approach also needs an extension to EKT in order to negotiate the SRTP Protection Profile used for end-to-end encryption and authentication. The RECOMMENDED default protection profile is AEAD_AES_128_GCM [I.D-draft-ietf-avtcore-srtp-aes-gcm].

The DTLS-SRTP procedures will result in the determination of an SRTP master key and master salt, along with an SRTP Protection Profile. This information is used for the hop-by-hop operations.

During the lifetime of a conference, the KMF MAY send a new EKT message to endpoints providing a new EKT key to use from that point forward, which might be desired when an endpoint leaves a conference for example.

If a new endpoint joins a conference and does not support the same SRTP Protection Profile in use, the KMF must initiate a new DTLS-SRTP handshake with all conference participants to negotiate a new security profile and to re-key the conference. This may cause some disruption to conference. Therefore, it is recommended that only a small number of protection profiles be required to implement by all endpoints.

To help in understanding better the sequence of messages and the relationship between the endpoint, MDD, and KMF, consider the following figure:

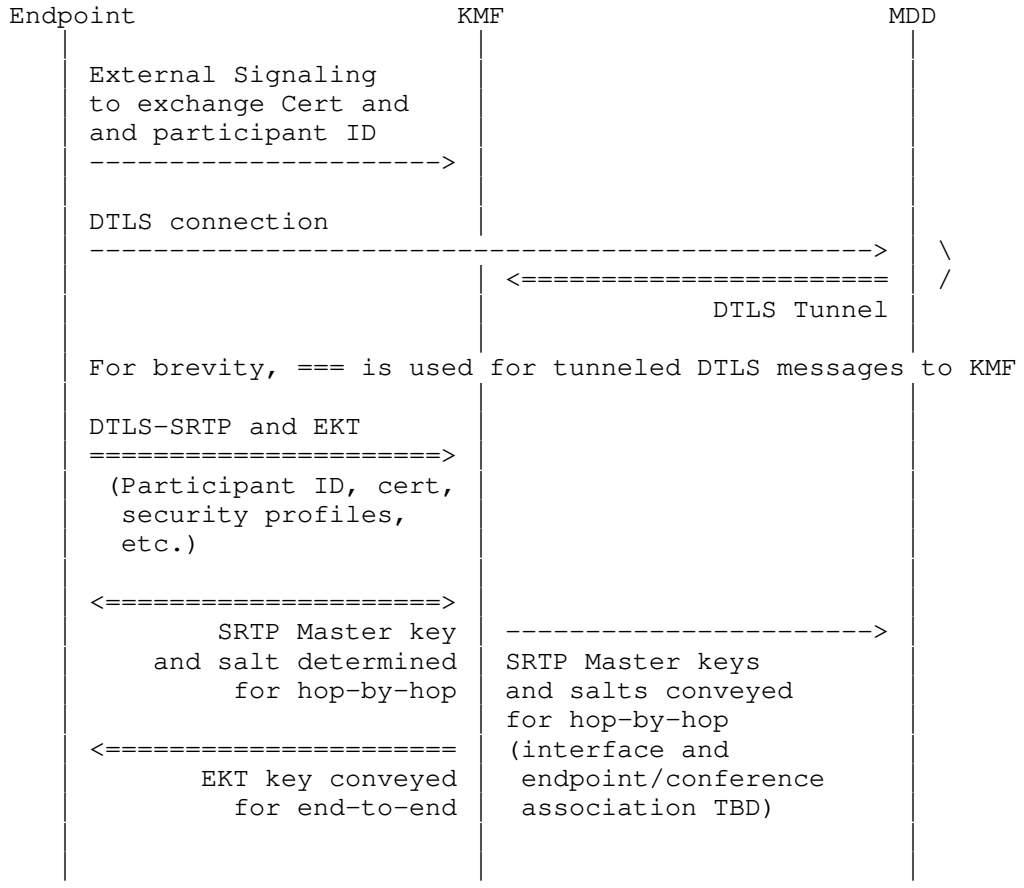


Figure 4 - Key Exchange Procedure

Following the key exchange, the endpoint will be able to encrypt media end-to-end and authenticate packets hop-by-hop. Likewise, the conference server will be able to authenticate the received packet at the hop, but will have no visibility into the encrypted media content.

8.2.2. MDD and KMF

The DTLS tunnel between the MDD and the KMF used to encapsulate the DTLS-SRTP signaling will also be used to convey the hop-by-hop encryption keys, salt, and protection profile information. In this way, no additional messages or interfaces are required in order for the switching MDD to receive the required security parameters.

9. Changing Media Forwarded and EKT Field

Endpoints transmit media to the MDD as they would to a traditional conference server, except that media is encrypted and authenticated with different keys as outlined in this framework. Each media source within an RTP session has a distinct SSRC and endpoints work to address SSRC collisions when they occur (see Section 8.2 of [RFC3550]). From the endpoint's perspective, what is particularly unique about the model described in this document is how the RTP payload (media content) is encrypted and authenticated end-to-end, while other security procedures are performed hop-by-hop.

To ensure a speedy decoder synchronization in receivers when transitioning from forwarding one active speaker's media to the next, a switching MDD will send a request for Full Intra-frame Request (FIR) [RFC5104] (also known as a "video fast update" in [H.323] systems) when a decision is made to switch active video flows. When the endpoint receives this request, it would transmit the video frame as requested and include with that initial packet the current "Full EKT Field" so that recipients will be able to decrypt the media flow. Additionally, a "Full EKT Field" should be transmitted about every 100ms to ensure that conference participants can decrypt the media transmitted.

It is not possible to request a "Full EKT Field" for audio flows. For this reason, it is RECOMMENDED that a "Full EKT Field" be included in audio packets about every 100ms to smooth the transition of the active speaker's audio forwarded by the server.

Endpoints SHOULD NOT include the "Full EKT Field" more frequently than specified herein, rather opting for the "Short EKT Field" when sending most packets to reduce the bandwidth consumed on the wire.

Endpoints MUST implement [RFC6464] in order for an MDD to determine which endpoint(s) have an active speaker as no other method requiring access to decrypted media can be used by an untrusted MDD.

10. To-Do List

10.1. What is needed to realize this Framework

- Endpoint must securely convey its certificate information to the KMF so the KMF can recognize a valid endpoint.

- A means through EKT or another mechanism to negotiate the SRTP security profiles for end-to-end encryption/authentication (e.g., proposing to negotiate AEAD_AES_128_GCM for end-to-end security)

- A means through EKT or another extension of sending the participant identifier (the participant identifier could

implicitly identify the conference) so the KMF will know which keys to provide for a given conference and RTP sessions related to that conference. Alternatively, this could be an element of the tunneling protocol, wherein the MDD indicates the associated identifiers.

- A change to EKT such that the ROC is transmitted in the clear, with integrity check performed by XORing the ROC with the IV used in AES Key Wrap

- A change to EKT to use MKI rather than ISN. This was proposed by John Mattsson during IETF 92 to address security issues and found to be extremely useful in easily managing multiple keys over a period of time. Use of MKI could be avoided if when a packet is received with a Full EKT Field, the key inside replacing any previously received key for that SSRC. However, in that case, the previous key would need to be retained for some period of time to handle out-of-order packets.

- A means of conveying per-hop SRTP master key and salt information to the switching MDD (which can be accomplished using the DTLS-SRTP tunneling protocol)

10.2. Other Considerations for this Framework

- Investigate adding ability to enable one-way media from a non-trusted device (e.g., announcements). While not specified as a requirement, it was mentioned during a previous IETF meeting and may be worth considering.

11. IANA Considerations

There are no IANA considerations for this document.

12. Security Considerations

[TBD]

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[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, May 2010.

[I.D-draft-ietf-avtcore-srtp-ekt] Mattson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for Secure RTP", Work in Progress, October 2014.

[RFC6904] J. Lennox, "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, December 2013.

[I.D-draft-ietf-avtcore-srtp-aes-gcm] McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in Secure RTP (SRTP)", Work in Progress, June 2015.

[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, May 2010.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

[RFC6464] Lennox, J., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, December 2011.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

[I.D-draft-abiggs-saag-key-management-service] Biggs, A. and S. Cooley, "Key Management Service Architecture", Work in Progress, July 2016.

13.2. Informative References

[I.D-draft-jones-perc-private-media-reqts] Jones, P. et al., "Private Media Requirements in Privacy Enhanced RTP Conferencing", Work in Progress, July 2015.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[H.323] Recommendation ITU-T H.323, "Packet-based multimedia communications systems", December 2009.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

14. Acknowledgments

The authors would like to thank Mo Zanaty and Christian Oien for invaluable input on this document.

Authors' Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

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

Nermeen Ismail
Cisco Systems, Inc.
170 W Tasman Dr.
San Jose
USA

Email: nermeen@cisco.com

David Benham
Cisco Systems, Inc.
170 W Tasman Dr.
San Jose
USA

Email: dbenham@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

P. Jones
D. Benham
Cisco

March 21, 2016

A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing
draft-jones-perc-private-media-framework-02

Abstract

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

Table of Contents

1. Introduction

Switched conferencing is an increasingly popular model for multimedia conferences with multiple participants using a combination of audio, video, text, and other media types. With this model, real-time media flows from conference participants are not mixed, transcoded,

transrated, recomposed, or otherwise manipulated by a media distribution device (MDD), 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 MDD 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 switching MDDs 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 MDDs can be deployed on general-purpose computing hardware. This, in turn, means that it is possible to deploy switching MDDs in virtualized environments, including private and public clouds. Deploying conference resources in a 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 the hardware. 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 privacy is ensured by making it impossible for an MDD 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 switching MDD 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 switching MDD by using a key that is independent from the media encryption and authentication key(s) and is unique to the participating endpoint and the switching MDD.

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

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

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

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

HBH (Hop-by-Hop): Communications between an endpoint and an MDD or between MDDs.

Endpoint: An RTP flow terminating entity that has possession of E2E media encryption keys and terminates end-to-end (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.

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

KMF (Key Management Function): An entity that is a logical function which passes keying material and related information to endpoints and MDDs that is appropriate for each. The KMF 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 MDDs.

Third Party: Any entity that is not an Endpoint, MDD, KMF 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.

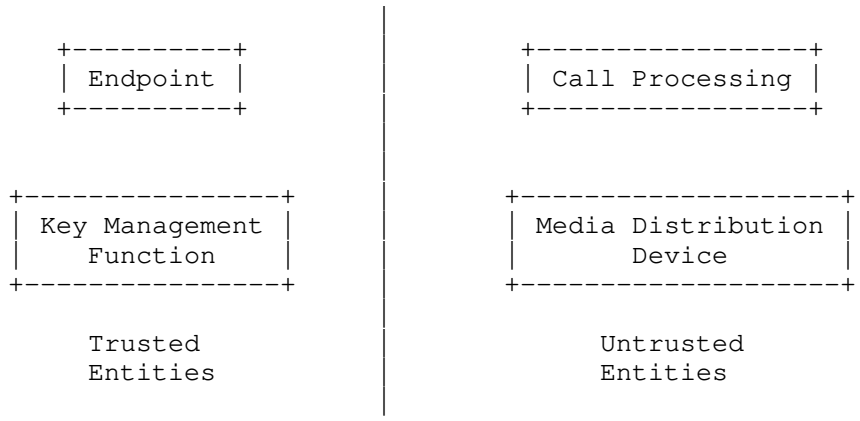


Figure 1: Trusted and Untrusted Entities in PERC

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. This does not mean that they are completely untrusted as they may be trusted with most non-media related aspects of hosting a conference.

3.1.1. MDD

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

An endpoint's ability to join a conference hosted by an MDD MUST NOT alone be interpreted as being authorized to have access to the E2E

media encryption keys, as the MDD does not have the ability to determine whether an endpoint is authorized.

An MDD 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 Section 6), etc, to a level equal to or better than traditional conferencing (i.e. pre-PERC) deployments.

An MDD 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 MDDs. Call processing may optionally ensure the privacy of call signaling messages between itself, the endpoint, and other entities.

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

3.2. Trusted Entities

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

3.2.1. Endpoint

An endpoint is considered trusted and will have access to E2E key information. While it is possible for an endpoint to be compromised, subsequently performing in undesired ways, defining endpoint resistance to compromise is outside the scope of this document. Endpoints will take measures to mitigate the adverse effects of denial of service attacks (refer to Section 6) from other entities, including from other endpoints, to a level equal to or better than traditional conference (i.e., pre-PERC) deployments.

3.2.2. KMF

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

Interaction between the KMF and the call processing function may be necessary to for proper conference-to-endpoint mappings, which may or may not be satisfied by getting information directly from the endpoints or via some other means. [TO DO: Revisit this text after design choice(s) are made between the alternatives.]

Obviously, the KMF needs to be closely managed to prevent exploitation by an adversary, as any kind of security compromise of the KMF puts the security of the conference at risk.

4. Framework for PERC

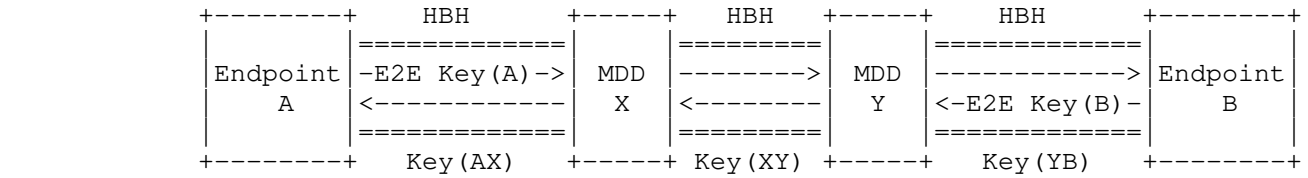
The purpose for this framework is to define a means through which media privacy can be ensured when communicating within a conferencing environment consisting of one or more MDDs that only switch, hence not terminate, media. It does not otherwise attempt to hide the fact that a conference between endpoints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

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

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

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

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



E2E and HBH Keys Used for Authenticated Encryption

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

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

4.2. E2E Key Confidentiality

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

Key / Entity	Endpoint A	MDD X	MDD Y	Endpoint B
KEK	Yes	No	No	Yes
E2E Key (i)	Yes	No	No	Yes
HBH Key (A<=>MDD X)	Yes	Yes	No	No
HBH Key (B<=>MDD Y)	No	No	Yes	Yes

Figure 2: Keys per Entity

4.3. E2E Keys and Endpoint Operations

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

Thus, endpoints MUST maintain a list of SSRCs from received RTP flows and each SSRC's associated E2E Key(i) information. Following a change of the KEK (i.e., EKT Key), prior E2E Key(i) information SHOULD be retained just long enough to ensure that late-arriving or out-of-order packets can be successfully decrypted and rendered. [NOTE: Perhaps a separate best practices document can recommend durations after some real world testing?] The endpoint SHOULD discard the E2E Key(i) and KEK information 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 MDD will not be able use the information contained in those header extensions with E2E encryption. [TO DO: Add a list to this doc of RTP Header Extensions that are off limits to - the MUST NOTs - be E2E encrypted.]

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 an MDD, as well between MDDs. The authentication key used for hop-by-hop authentication is derived from an SRTP master key shared only on the respective hop (HBH Key(j); j={a given hop}). Each HBH Key(j) is distinct per hop and no two hops ever intentionally use the same SRTP master key.

Using hop-by-hop authentication gives the MDD the ability to change certain RTP header values. Which values the MDD can change in the RTP header are defined in [I-D.jennings-perc-double]. RTCP can only be encrtpted, giving the MDD 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 switching MDD visibility into header extensions, such as the one used to determine audio level [RFC6464] of conference

participants. Note that when RTP header extensions are encrypted, all hops - in the untrusted domain at least - will need to decrypt and re-encrypt these encrypted header extensions.

4.5. Key Exchange

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

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

4.5.1. Initial Key Exchange and KMF

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

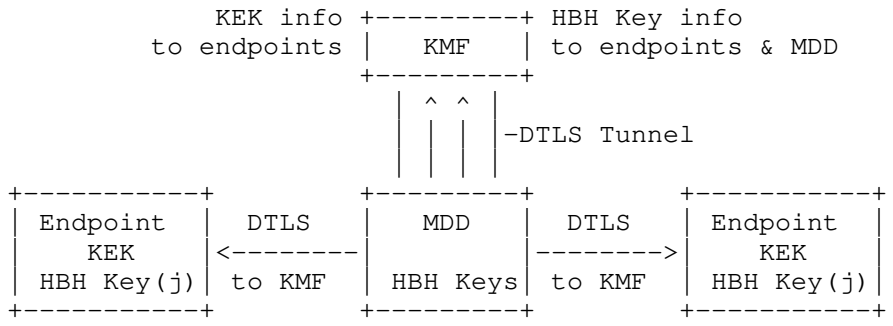


Figure 3: Exchanging Key Information Between Entities

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

4.5.2. Key Exchange during a Conference

Following the initial key information exchange with the KMF, endpoints will be able to encrypt media end-to-end with their E2E Key(i), sending that E2E Key(i) to other endpoints encrypted with KEK, and will be able to encrypt and authenticate RTP packets using local HBH Key(j). The procedures defined do not allow the MDD 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 KMF decides to rekey a conference, it transmits a specific message defined in PERC EKT [I-D.jennings-perc-srtp-ekt-diet] to each of the conference participants. The endpoint MUST create a new SRTP master key and prepare to send that key inside a Full EKT Field using the new EKT Key. Since it may take some time for all of the endpoints in conference to finish re-keying, senders SHOULD 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. [TO DO: Either need to pick a short delay period for endpoints to use per above, defer to a future best practices document or consider having the KMF manage the delay period given it knows the size of a given conference.]

5. Entity Trust

It is important to this solution framework that the entities can trust and validate the authenticity of other entities, especially the KMF 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 KMF for the conference and the KMF 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 (EDIT: add reference) Identity assertion 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 KMF to ensure that only authorized users participate in the conference. Similarly the KMF can create a WeBRTC Identity assertion bound the fingerprint of the unique certificate used by the KMF for this conference so that the endpoint can validate it is talking to the correct KMF.

5.2. Certificate Fingerprints in Session Signaling

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

As a concrete example, SIP [RFC3261] and SDP [RFC4566] can be used to convey the fingerprint information per [RFC5763]. An endpoint's SIP User Agent would send an INVITE message containing SDP for the media session along with the endpoint's certificate fingerprint, which can be signed using the procedures described in [RFC4474] for the benefit of forwarding the message to other entities. Other entities can now verify the fingerprints match the certificates found in the DTLS-SRTP connections to find the identity of the far end of the DTLS-SRTP connection and check that is the authorized entity.

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

6. Attacks on Privacy Enhanced RTP Conferencing

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

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

6.1. Third Party Attacks

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

Off-path attackers may try connecting to different PERC entities and send specifically crafted packets. A successful attacker might be able to get the MDD to forward such packets. If not making use of HBH authentication on the MDD, 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 an MDD, fooling endpoints in to sending packets to the false MDD 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 MDD may cascade to another legitimate MDD creating a false version of the real conference. This attack is mitigated since false MDD would not be authenicated by the KMF and be able tunnel the DTLS-SRTP signaling to/from the KMF. Since it cannot tunnel the DTLS-SRTP signaling, endpoints will not get the KEK and the conference will not go on. [TO DO: Include the exchange valid MDD certficates with the KMF in this doc or not?]

6.2. MDD Attacks

The MDD 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 MDD 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 MDD can attempt to consume the receiving endpoints resources.

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

7. To-Do List

7.1. What is Needed to Realize this Framework

- Endpoints and KMF must securely convey their respective certificate information directly or indirectly via some other means or identity service provider.

- If as in "Double" draft, the ROC value is no longer in the clear and associated with the "outer" protection scheme, we may need to require that the MDD maintain a separate ROC value for each SSRC sent to each separate endpoint. This ROC value should start at 0 regardless of the sequence number in that first packet sent to an endpoint. [EDIT: Do we document this in this framework or in Double draft?]

- Investigate adding ability to enable the transmission of one-way media from a non-trusted device (e.g., announcements). One possible solution is to have the KMF send an "ekt_key" message that is explicitly labeled for receive-only and giving that to announcement servers. As opposed to modifying the EKT spec for this PERC-specific need, we could say in the framework that EKT Keys with a SPI > 32000, say, are intended for this purpose and trusted endpoints should only use those EKT Keys to decrypt Full EKT Fields received from such transmitters. Thus, trusted endpoints would never send media with EKT Keys having those SPI values.

8. IANA Considerations

There are no IANA considerations for this document.

9. Security Considerations

[TBD]

10. Acknowledgments

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

11. References

11.1. Normative References

[I-D.jennings-perc-double]
Jennings, C., Jones, P., and A. Roach, "SRTP Double Encryption Procedures", draft-jennings-perc-double-01 (work in progress), March 2016.

[I-D.jennings-perc-srtp-ekt-diet]
Mattsson, J., McGrew, D., Wing, D., Andreasen, F., and C. Jennings, "Encrypted Key Transport for Secure RTP", March 2016.

[I-D.jones-perc-dtls-tunnel]
Jones, P., "DTLS Tunnel between Media Distribution Device and Key Management Function to Facilitate Key Exchange", draft-jones-perc-dtls-tunnel-02 (work in progress), March 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

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

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

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

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

11.2. Informative References

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

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

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

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

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

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

Authors' Addresses

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

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

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

Email: dbenham@cisco.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: January 6, 2016

P. Jones (Ed.)
N. Ismail
D. Benham
N. Buckles
Cisco Systems
J. Mattsson
Ericsson
R. Barnes
Mozilla

July 6, 2015

Private Media Requirements in Privacy Enhanced RTP Conferencing
draft-jones-perc-private-media-reqts-00

Abstract

This document specifies the requirements for ensuring the privacy and integrity of real-time transport protocol (RTP) media flows between two or more endpoints communicating through one or more centrally located media distribution devices (MDDs).

Status of this Memo

This Internet-Draft is submitted to IETF 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 http://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 January 6, 2015.

Table of Contents

1. Introduction

Users of multimedia communication products and services have privacy expectations that are largely satisfied with the use of SRTP [RFC3711] and related technologies when communicating point-to-point over the Internet. When two or more endpoints communicate through a traditional media server, it is necessary for those endpoints to share the SRTP master key and salt information with the traditional media server so that it can authenticate and decrypt received RTP and RTCP packets. The key material is needed so that a traditional media server can perform various operations on the media, such as mixing,

transcoding, and transrating. The traditional media server also needs the master key and salt in order to transmit media packets to other endpoints in the conference. The need for a traditional media server to have the master key represents a security risk.

Within a corporate or other isolated environment where all conferencing resources, including both call control and media processing functions, are tightly controlled, this security risk can be effectively managed. However, managing this risk is becoming increasing difficult as conferencing resources are deployed in networks that are not so strictly managed or controlled, including resources on virtualized servers deployed in third-party cloud environments.

There are also existing public voice and video conferencing service providers in which users must place full trust by sharing media encryption keys in order to use those services. This exposes corporations, for example, to a higher risk of being subjected to corporate espionage. While it is not the intent of this draft to suggest that any existing service provider would permit or condone any illicit use of its service, the fact is that security threats can come from either internal or external sources and remain undiscovered for long periods of time.

It is possible to ensure real-time transport protocol (RTP) media privacy in deployments using one or more centrally located media distribution devices (MDDs) with limited changes in the security mechanisms used today. This document discusses this possibility in more detail and presents a set of requirements that are neutral with respect to session signaling protocols.

This document is focused on ensuring the privacy of RTP media in centralized MDD models only. Other types of media are out of scope. Other, non-centralized media distribution models are also out of scope.

2. Requirements Language

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 RFC 2119 [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.

3. Terminology

Adversary - An unauthorized entity that may attempt to compromise the performance of a media distribution device through various means, including, but not limited to, the transmission of bogus media packets or attempt to gain access to the plaintext of the media.

Media content - The portion of the RTP (i.e., the encrypted RTP payload) or other packet containing the actual audio, video, or other multimedia information that is considered confidential and is subject to end-to-end encryption. This does not include, for example, RTP headers, RTP header extensions, or RTCP packets.

Switching media distribution device - A media distribution device that does not decrypt RTP media flows or perform processing on the media payload, but instead simply forwards the received media from a sender to the other endpoints in a multimedia conference. A switching media distribution device may modify some portion of the RTP header and may often consume and create RTCP messages for efficient media handling.

4. Background

Traditional media servers used for multimedia conferencing would mix, transcode, transrate, and/or recompose media flows from one or more conference participants' endpoints, sending out a different audio and video flow to each endpoint. For audio, this might entail mixing some number of input flows that appear to contain audio intended to be heard by the other participants, with each endpoint receiving a flow that does not contain that participant's own audio. For video, the traditional media server may elect to send only video showing the current active speaker, a tiled composition of all participants or the most recent active speakers, a video flow with the active speaker presented prominently with other participants presented as thumbnail images, or some other composite arrangement. It is also common for audio or video to be transcoded. A typical traditional media server is depicted in Figure 1.

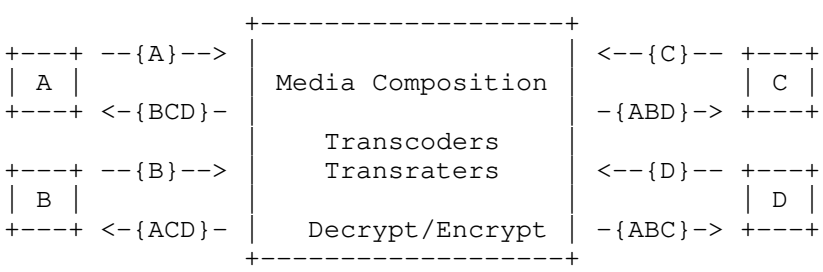


Figure 1 - Traditional Media Server

Traditional media servers require a significant amount of processing power, which in turn translates into a high cost for conferencing hardware manufacturers. Significantly, too, it is very difficult to deploy these servers in a cloud environment due to the high processing demands, as the specialized hardware found in the traditional media server does not exist in a cloud environment.

To enable the traditional media server to perform its job, the server establishes one or more SRTP sessions with each of the conference endpoints wherein it is given access to the keys required to decrypt and encrypt media flows from and to each endpoint. This means that the traditional media server is necessarily a fully trusted entity in the communication path. Any time these servers are deployed in a network that is not secured, it increases the risk that an adversary might gain access to cryptographic key material, allowing the adversary to be able to see and listen to ongoing conferences. In some instances, depending on how the hardware is designed and how keys and certificates are managed, it might be possible for an adversary to see and listen to previously recorded conferences or future conferences.

The Secure Real-time Transport Protocol (SRTP) [RFC3711] is a profile of RTP, which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the RTP Control Protocol (RTCP). Encryption of header extension in SRTP [RFC6904] provides a mechanism extending the mechanisms of [RFC3711], to selectively encrypt RTP header extensions in SRTP. [RFC3711] and [RFC6904] solves end-to-end use cases between two endpoints, and does not consider use cases where a sender delivers media to a receiver via a cloud-based conferencing service.

5. Motivation for Private Media using switching MDDs

5.1. Switching Media in Cloud Services

There is a trend in the industry for enterprises to use cloud services to host multi-party conferences and meet-me services, either exclusively or to meet peak loads on-demand. At the same time, there is shift toward using lightweight, cost-effective switching MDDs in cloud services that do not necessarily need to mix audio or composite/transcode video. Also fueling the use of such lightweight MDDs is the desire to fully exploit virtualized computing resources and dynamic scalability potential available in cloud computing environments.

The increased use of cloud services has exposed a problem. There are two different trust domains from a media perspective: endpoints and other devices in a trusted domain, and MDDs controlled by the cloud service in an untrusted domain. Other examples of conference devices spread across trusted and untrusted domains are likely, but the cloud service trend is triggering the urgency to address the need to allow for lightweight media conference while enabling media privacy at the same time.

With a switching MDD, each endpoint transmits media as it would with a traditional media server. However, the switching MDD merely forwards all or a subset of the media to the other endpoints in the conference (where at least one other endpoint may be associated with

a cascaded media distribution device), leaving composition to the receiving endpoint. It is also worth noting that, for a switching MDD model to work successfully, each endpoint in the conference must support the media formats transmitted by all other endpoints in the conference. More modern endpoints support multiple codecs and formats, making this commercially practical.

Figure 2 depicts an example of a switching MDD wherein each endpoint is receiving the media flows transmitted by each of the other endpoints in the conference.

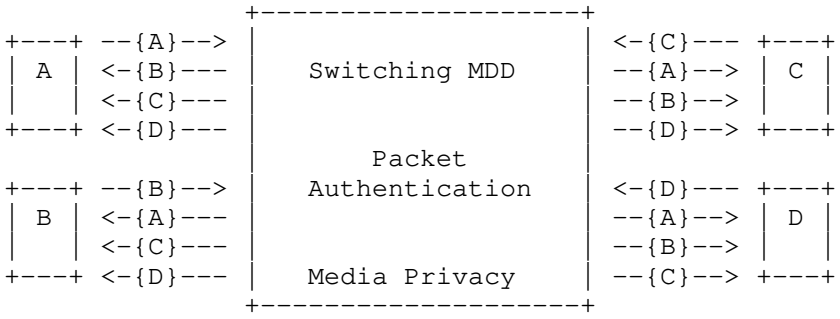


Figure 2 - Switching Media Distribution Device

Note - The use of multiple arrows directed toward each endpoint is not intended to suggest the use of separate RTP sessions.

By using methods such as those described in [RFC6464], it is possible for the switching MDD to transmit the appropriate audio and video flows to endpoints without having knowledge of the content of the encrypted media. The following "Active Speaker Switching" examples help illustrate this point.

In Figure 3, endpoints A, B and D receive the video streams from endpoint C, the currently active speaker, which is receiving video from endpoint A, the previous active speaker. Later when endpoint B becomes the active speaker (Figure 4), endpoints A, C and D will start to receive video from B, while endpoint B continues to receive video from endpoint C. Finally in Figure 5, endpoint A becomes the active speaker.

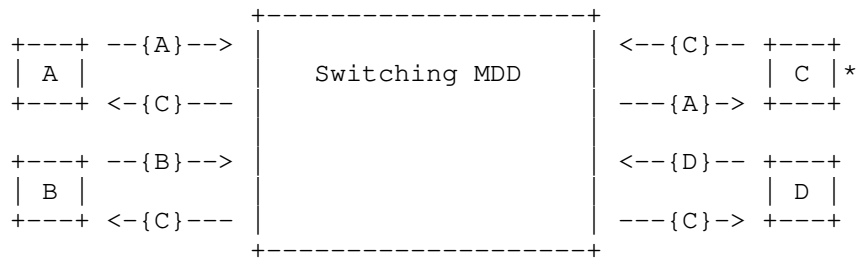


Figure 3 - Endpoint "C" is the Active Speaker

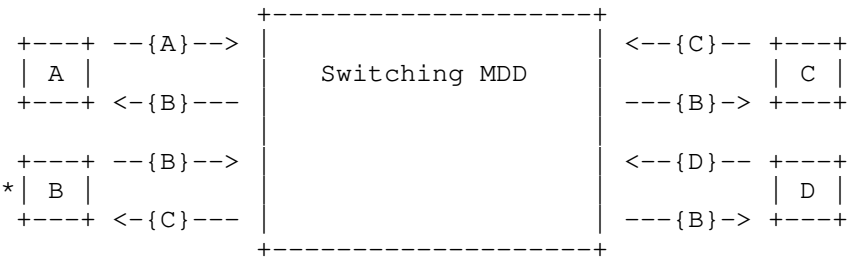


Figure 4 - Endpoint "B" is the Active Speaker

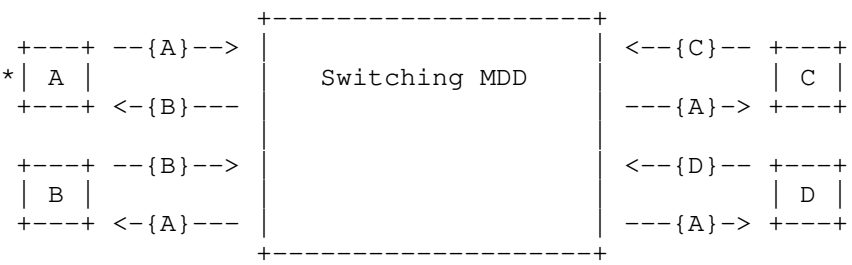


Figure 5 - Endpoint "A" is the Active Speaker

Switched media can also enable conferences to scale to include many more endpoints simultaneously than would be possible with a traditional media server. Like traditional media servers, switching MDDs can also be cascaded or interconnected in a meshed topology to increase the size of the conference without putting undue burden on any particular server.

5.2. Private Media Security through Switching

A traditional media server, or MCU, establishes an SRTP session with each endpoint separately, and needs to decrypt packets containing media for presentation to other endpoints. By using a switching MDD, it is possible to keep the media encryption keys private to the endpoints such that the MDD does not have access to the keys used for

media encryption. The switching MDD just forwards media received to each of the other endpoints in the conference.

This provides for a significantly improved security model, as one can, for example, utilize conferencing resources in the cloud that do not have to be trusted. That said, there may be situations where the switching MDD needs to modify the RTP packet received from an endpoint, such as by adding or removing an RTP header extension, modifying the payload type value, etc. It would be the responsibility of the switching MDD to ensure that media of the expected type and containing the correct information is received by a recipient.

Thus, there is a need to utilize an end-to-end encryption and authentication key (or pair of keys) and a hop-by-hop encryption and authentication key (or pair of keys). The end-to-end encryption and authentication key(s) is to ensure that media remains private to the trusted endpoints. The hop-by-hop authentication key allows the switching MDD to authenticate RTP and RTCP packets and to optionally modify certain elements of those packet. The hop-by-hop encryption key is to optionally encrypt RTP header extensions and optionally encrypt RTCP packets. The current SRTP and related specifications do not define use of a dual-key (hop-by-hop and end-to-end) approach. However, such an approach is possible and would result in ensuring the privacy of media while also enabling the more scalable switched conferencing model.

This dual-key model does necessitate a change in the way that keys are managed. However, the topic of key management is outside the scope of this requirements document. High-level assumptions, such as if the end-to-end context uses a group key as SRTP master key or if individual SRTP master keys (that may be derived/negotiated from another group key), are likely to influence the solution derived from this document.

6. Private Media Trust Model

The architectural model suggested in this document enables switching MDDs to be hosted in domains in which the network elements may have low trust, or where the trustworthiness is uncertain. This does not mean that the service provider is completely untrusted; it simply means that high enough trust with media decryption is not required. This has the benefit of protecting the endpoint's media in the case of external attacks against the MDD.

In this model, certain elements are considered trusted and others are considered untrusted. Trust in the context of this document means that the element can be in possession of the media encryption key(s) for a past, current, or potentially future conference (or portion thereof) used to protect media content.

In the general case, only the endpoint and an associated key management function, which may be integrated with the endpoint or in a separate stand-alone entity, needs to be trusted. However, it is recognized that in certain deployments, some elements that are classified as untrusted in this document might be placed into the trusted domain and thus be considered trusted. One example might be a gateway, traditional media server or other MDD in a trusted environment connecting endpoints to the same private media conference. This document does not preclude such deployment combinations, but does not rely on them in order to keep the examples and model definitions focused on the simple, most general case.

Each of the elements discussed below has a direct or indirect relationship with each other. The following diagram depicts the trust relationships described in the following sub-sections and the media or signaling interfaces that exist between them, showing the trusted elements on the left and untrusted elements on the right. Note that this is a functional diagram and elements may be co-located or further divided into multiple separate physical entities. Further, it is not necessary that every interface exist between all elements, such as both an interface from the endpoint and call processing function to a key management function, though both are possible options.

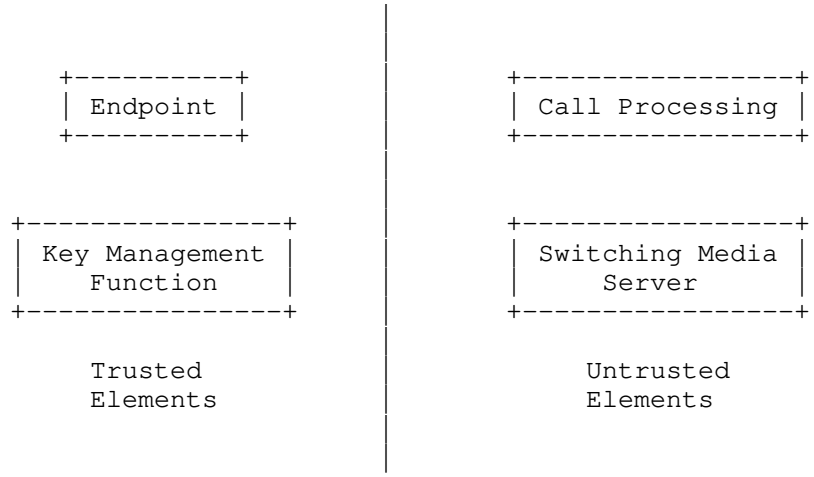


Figure 6 - Relationship of Trusted and Untrusted Elements

6.1. Trusted Elements

The endpoint is considered a trusted element, as it will be sourcing media flows transmitted to other endpoints and will be receiving media for rendering. While it is possible for an endpoint to be compromised and perform in unexpected ways, such as transmitting a decrypted copy of media content to an adversary, such security issues and defenses are outside the scope of this document.

The other trusted element is a key management function (KMF), which may be integrated with the endpoints or exist standalone. This function is responsible for providing cryptographic keys to the endpoints for encrypting and authenticating media content. The KMF is also responsible for providing cryptographic keys to the conferencing resources, such as the MDD, to enable authentication of media packets received by an endpoint. Interaction between the KMF and untrusted call processing functions may be necessary to ensure endpoints are delivered the appropriate keys. The KMF needs to be tightly controlled and managed to prevent exploitation by an adversary, as any kind of security compromise of the KMF puts the security of the conference at risk.

6.2. Untrusted Elements

The call processing function is responsible for such things as authenticating the user or endpoint for the purpose of joining a conference, signing messages, and processing call signaling messages. This element is responsible for ensuring the integrity, and optionally the confidentiality, of call signaling messages between itself, the endpoint, and other network elements. However, it is considered an untrusted element for the purposes of this document, as it cannot be trusted to have access to or be able to gain access to cryptographic key material that provides privacy and integrity of media packets.

There might be several independent call processing functions within an enterprise, service provider network, or the Internet that are classified as untrusted. Any signaling information that passes through these untrusted entities is subject to inspection by that element and might be altered by an adversary.

Likewise, there may be certain deployment models where the call processing function is considered trusted. In such cases, trusted call processing functions MUST take responsibility for ensuring the integrity of received messages before delivering those to the endpoint. How signaling message integrity is ensured is outside the scope of this document, but might use such methods as defined in [RFC4474].

The final element is the switching MDD, which is responsible for forwarding encrypted media packets and conference control information to endpoints in the conference. It is also responsible for conveying secured signaling between the endpoints and the key management function, acquiring per-hop authentication keys from the KMF, and performing per-hop authentication operations for media packets. This function might also aggregate conference control information and initiate various conference control requests. Forwarding of media packets requires that the switching MDD have access to RTP headers or header extensions and potentially modify those message elements, but

the actual media content MUST not be decipherable by the switching MDD.

Further, the switching MDD does not have the ability to determine whether an endpoint is authorized to have access to media encryption keys. Merely joining a conference MUST NOT be interpreted as having authority. Media encryption keys are conveyed to the endpoint by the KMF in such a way as to prevent the switching MDD from having access to those keys.

It is assumed that an adversary might have access to the switching MDD and have the ability to read any of the contents that pass through. For this reason, it is untrusted to have access to the media encryption keys.

As with the call processing functions, it is appreciated that there may be some deployments wherein the switching MDD is trusted. However, for the purposes of this document, the switching MDD is considered untrusted so that we can be ensure to develop a solution that will work even in the most hostile environments.

It is expected that a switching MDD performs its role in properly forwarding media packets, taking measures to safeguard against replay attacks, etc. If a MDD is exploited, an adversary may do such things as discard packets, replay packets, or introduce unacceptable delay in packet delivery.

7. Goals and Non-Goals

7.1. Goals

7.1.1. Ensure End-To-End Confidentiality

The content of the communication and all media needs to be confidential within the group of entities explicitly invited into the conference. An external monitoring adversary should not be able to deduce the human-to-human communication that actually occurred from capturing the media packets.

At the same time, it is necessary to allow switching MDDs to manipulate certain RTP header fields like the payload type value.

7.1.2. Ensure End-To-End Source Authentication of Media

In a conference system with multiple endpoints it is vital that the media content presented to any of the human participants is from the stated endpoint, and not an adversary that attempts to inject misleading content. Nor should an adversary be able to fool the system into becoming a trusted party in the conference. Only explicitly invited parties shall be able to contribute content.

7.1.3. Provide a More Efficient Service than "Full-Mesh"

A multi-party conference that has the goals of confidentiality and source authentication can be established as a "full mesh" (i.e., each participating endpoint directly addresses each of the other endpoints). However, this has a significant issue with the amount of consumed resources in both the uplink and the downlink from each endpoint.

A switched conferencing model would yield the efficiencies desired.

7.1.4. Support Cloud-Based Conferencing

To achieve cost-effective and scalable conferencing, it must be possible to run the MDD instances in a cloud-based virtualized environment.

From a security standpoint, this is a significant issue since the virtualized server instance and the underlying hardware and software upon which it runs might not be secure from an adversary.

7.1.5. Limiting an Endpoint's Access to Content

Since an invited endpoint will be provided with the content protection keys, the endpoint can decrypt content from time periods before and after the endpoint joined the conference. However, this is not always desirable. It should be possible to re-key the content protection keys every time a participant joins or leaves the conference so each particular set of endpoints uses a unique key.

This also changes the trust level required on the conference roster handling at any point and how to keep that accurate and secured.

It should be noted that timely completion of the re-keying operations become an obstacle in system design and operation. Thus, it is a goal to allow for this possibility when it is deemed essential, but it should not be a requirement on a system to re-key each time the participant list changes.

7.1.6. Compatibility with the WebRTC Security Architecture

It is a goal of this work to ensure compatibility with the WebRTC security architecture as described in [I.D-rtcweb-security-arch]. As an example, local resources that are considered a part of the trusted computing base (TCB), such as keying material derived using DTLS-SRTP, will remain within the TCB and not exposed to untrusted entities.

The browser is reliant on an external calling service to convey signaling information that may open the door for a man-in-the-middle attack, such as the conveyance of certificate fingerprints over the

interface between the browser and the calling service. However, as described in [I.D-rtcweb-security-arch], the browser may utilize additional services, such as a trusted identify provider, to mitigate such risks.

Having said the foregoing, this document does not aim to define requirements for end-to-end security for the WebRTC data channel.

7.2. Non-Goals

7.2.1. Securing the Endpoints

The security of a communication session requires that the endpoints are not compromised and that the users are trustworthy. If not, credentials and decrypted content may be shared with third parties. However, this is hard to prevent through system design. Thus, it should be assumed that the endpoint is secure and the user is trustworthy; how to achieve this is out of scope this document.

7.2.2. Concealing that Communication Occurs

A non-goal is to attempt to prevent a pervasive monitoring adversary from knowing that the communication session has occurred. The reason for excluding this as a goal is that it is extremely difficult to achieve, as a pervasive monitoring adversary can be expected to be able to have knowledge of all IP flows that enter or exit local ISPs, across links that straddle national borders or internet exchange points. To hide the fact communication occurred, the flows required to achieve the communication session need to be highly difficult to correlate between different legs of the communication.

At this stage this is deemed too difficult to attempt and will need to be a subject for further study. Existing attempts include The Onion Router (TOR), against which it has been claimed to be possible to monitor, at least partially, by an adversary with sufficient reach.

Also of consideration is that trying to conceal the fact that communication occurred actually makes it more difficult for network administrators to effectively manage and troubleshoot issues with conference calls.

7.2.3. Individual Media Source Authentication

Although the endpoints in the conference are authenticated, it is not a goal to provide source authentication of the media at the individual user level, instead being satisfied with being able to authenticate media as coming from an invited endpoint or not.

There exist solutions that can provide individual media source authentication (e.g., TESLA). However, they impact the performance

or security properties they provide. Thus, further study is required to determine impact and resulting security properties if desired to have individual source authentication.

7.2.4. Multicast -based Conferencing

Using multicast to construct a non-centralized media distribution model is out of scope. This document is focused only on models where endpoints, or other devices, participating in a conference unicast media to a centrally located media distribution device.

8. Requirements

The following are the security solution requirements for switched conferencing that enable end-to-end media privacy between all endpoints.

Note that while some switching MDDs might be fully trusted entities, the intent of this solution and purpose for these requirements is to address those servers that are not trusted.

PM-01: Switching media distribution device MUST be able to switch the media between endpoints in a conference without having access to unencrypted media content.

PM-02: Solution MUST maintain all current SRTP security goals, namely the ability to provide for end-to-end confidentiality, provide for hop-by-hop replay protection, and ensure hop-by-hop and end-to-end message integrity.

PM-03: Solution MUST extend replay protection to cover each hop in the media path, both ensuring that any received packet is destined for the recipient and not a duplicate.

PM-04: Keys used for end-to-end encryption and authentication of RTP payloads and other information deemed unsuitable for access by the switching media distribution device MUST NOT be generated by or accessible to any component that is not trusted.

PM-05: The switching media distribution device MUST be allowed to make changes to the RTP header and the RTP header extensions.

PM-06: A cryptographic context suitable for enabling end-to-end authenticated encryption MUST be defined.

PM-07: The switching media distribution device, or any entity that is not fully trusted, MUST NOT be involved in the user or endpoint authentication for the purpose of media key distribution.

PM-08: The switching media distribution device MUST be able to switch an already active RTP stream to a new receiver, while guaranteeing the timely synchronization between the RTP security context of the transmitter and its current and new receivers.

PM-09: It MUST be possible for the switching media distribution device to determine if a received media packet was transmitted by an endpoint in possession of a valid hop-by-hop key for that conference.

PM-10: It MUST be possible for a conference to be optionally re-keyed as desired, such as each time a participant joins or leaves the conference.

PM-11: Any solution satisfying this requirements document MUST provide for a means through which WebRTC-compliant endpoints can participate in a switched conference using private media as outlined herein.

PM-12: All RTP senders, including the switching media distribution device, MUST adhere to all congestion control requirements that are required by the RTP profile and topology in use, including RTP circuit breakers [I.D-ietf-avtcore-rtp-circuit-breakers]. Since the switching media distribution device is unable to perform transcoding or transrating that requires access to the unencrypted media, its reaction to congestion signals is often limited to dropping packets that would otherwise be forwarded in the absence of congestion, and signaling congestion to the RTP source. This is similar to the congestion control behavior of the Media Switching Mixer and Selective Forwarding Middlebox/Unit in [I.D-ietf-avtcore-rtp-topologies-update].

PM-13: It MUST be possible for a media distribution device or an endpoint to authenticate a received RTCP packet.

9. IANA Considerations

There are no IANA considerations for this document.

10. Security Considerations

[TBD]

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC6464] Lennox, J., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, December 2011.

[I.D-rtcweb-security-arch] E. Rescorla, "WebRTC Security Architecture", Work in Progress, March 2015.

[RFC6904] J. Lennox, "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, December 2013.

[I.D-ietf-avtcore-rtp-topologies-update] Westerlund, M., and S. Wenger, "RTP Topologies", Work in Progress, March 2015.

[I.D-ietf-avtcore-rtp-circuit-breakers] Perkins, C. S., and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", Work in Progress, March 2015.

11.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

12. Acknowledgments

The authors would like to thank Marcello Caramma, Matthew Miller, Christian Oien, Magnus Westerlund, Cullen Jennings, Christer Holmberg, Bo Burman, Jonathan Lennox, Suhas Nandakumar, Dan Wing, Roni Even, and Mo Zanaty for their invaluable input.

13. Contributors

Yi Cheng
Ericsson
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 17 589
Email: yi.cheng@ericsson.com

Authors' Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

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

Nermeen Ismail
Cisco Systems, Inc.
170 W Tasman Dr.
San Jose
USA

Email: nermeen@cisco.com

David Benham
Cisco Systems, Inc.
170 W Tasman Dr.
San Jose
USA

Email: dbenham@cisco.com

Nathan Buckles
Cisco Systems, Inc.
170 W Tasman Dr.
San Jose
USA

Email: nbuckles@cisco.com

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

Richard Barnes
Mozilla
331 E Evelyn Ave.

Mountain View
USA

Email: rlb@ipv.sx

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

J. Mattsson
M. Naslund
Ericsson

July 6, 2015

Secure Real-time Transport Protocol (SRTP) for Cloud Services
draft-mattsson-perc-srtp-cloud-00

Abstract

In order to support use cases when two or more end-points communicate via one (or more) cloud service (e.g. virtualized cloud-based conferencing) that are not trusted to access the media content, this document describes the use of so called end-to-end (inner) and hop-by-hop (outer) cryptographic transforms within the Secure Real-time Transport Protocol (SRTP). One of the main aspects of the transforms is to make the confidentiality and message authentication independent of the RTP header. Another central aspect is to enable identification of the cryptographic contexts (keys etc.). Besides the security of the end-points, also trust assumptions regarding the cloud services are addressed.

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 http://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 January 7, 2016.

Table of Contents

1. Introduction

The Secure Real-time Transport Protocol (SRTP) [RFC3711] is a profile of RTP, which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the RTP Control Protocol (RTCP). The basic SRTP profile in [RFC3711] addresses real-time end-to-end use cases, and does not consider use cases where a sender delivers media to one or more receivers via a cloud-based service. One typical example of such use cases is multi-party conferencing, where a middlebox (conference server) forwards media received from one participant to all other participants. Figure 1 below shows a conference with four participants.

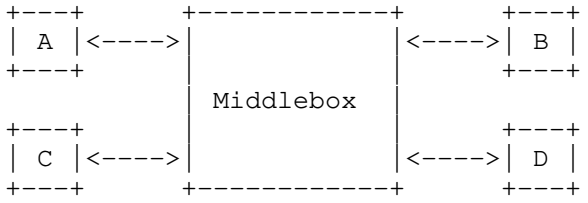


Figure 1: Multi-party conference with a middlebox

The cloud based middlebox is typically considered as semi-trusted, meaning that the middlebox will deliver media as requested, but it cannot be excluded that the middlebox will also try to extract the information in the media(e.g. infringement of copyrighted content, legal or illegal intercept). The reason to not use a fully trusted middlebox is mainly cost and convenience, the same forces that drives out-sourcing and cloud computing. The trust model will be made more formal later in this document. What causes problems for standard end-to-end SRTP in these settings is its dependence on the actual RTP transport parameters which will differ when RTP is used on different hops, i.e., sender-middlebox and middlebox-receiver.

SRTP is a framework that allows new security functions and new transforms to be added and this document defines extensions to SRTP to meet the additional use cases considered. One of the main aspects

of the transform is to make the confidentiality and message authentication independent of the RTP header. This allows for end-to-end protection to be achieved also when cloud based middleboxes assign values to the RTP headers, independently on each hop.

Another aspect is that identification of the cryptographic context (keys etc.) between the end-points must be extended, as the parameters used in [RFC3711] are available only during transport of RTP packets over a "hop". For instance, [RFC3711] specifies that the receiver's IP address shall be part of the context identifier, but this value may of course not be known to the sender when communicating messages via a cloud based middlebox. One may also want to avoid using IP address as context identifier due to privacy considerations. Another part of the cryptographic context identifier is the SSRC, which may be modified by middleboxes.

While there certainly are differences between this document and [RFC3711] on mechanism level, it is worth noticing that the kind of extensions defined herein are conceptually almost identical to the SRTP extensions previously defined in [RFC4383], which adds source origin authentication support to SRTP. Moreover, as far as the cryptographic processing is concerned, the cloud based middleboxes may use [RFC3711] compliant processing and changes in cryptographic processing are thus only needed in the end-points.

1.1. Scope of this Document

The scope of this document is to specify extensions to SRTP (parameters, processing, and cryptographic transforms) to support use cases where a cloud based middlebox is involved and to describe the associated trust model.

The existence of cloud based middlebox implies a different trust model than that originally considered when designing SRTP. This manifests itself in terms of the need to ensure only authorized access to the different cryptographic keys involved, i.e. the extensions defined herein MUST have support from some key management scheme. Similar to the original SRTP specification, the actual definition of the key management solution is out of scope of this document.

1.2. 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].

Definitions:

DoS: Denial of service

e2e: end-to-end

hbh: hop-by-hop

2. SRTP Cloud Overview

2.1. Overview

A high-level description of the proposed new SRTP functionality is as follows: The first step is to perform a transport independent media protection operation. The coverage of this transform is the RTP payload only. This operation is done with an Authenticated Encryption with Associated Data (AEAD) transform. The media protection should rely on two explicit values for cryptographic synchronization, the Packet Unique Value (PUV) and the SRTP Source (SSS), which are included and forwarded in the payload.

After the steps making up the transport independent media protection have been performed, the protection processing proceeds as currently defined by [RFC3711], which results in the addition of the required transport protection.

Keying for transport protection is performed as described in [RFC3711] and uses the SRTP internal key derivation function. The key derivation function operates on a master key and a master salt, where the master key (and salt) is here denoted hbh key (and hbh salt).

The keying for the media protection is defined in an equivalent way, producing keying material for the media transform. The e2e keying material is based on another master key, the e2e key, which is independent of the hbh key. Also for the e2e context, a master salt (e2e salt) is defined. The key derivations used to derive the e2e keying material could preferable use the key derivation function defined in [RFC3711].

Note that with the approach taken, only the media protection endpoints will have to implement the handling of two security contexts. One of the defined transforms of [RFC3711] is used for the transport protection (using the hbh key). A cloud middlebox should be able to reuse a [RFC3711] compliant implementation of SRTP to first receive and then resend the media.

For RTCP, the solution principles described for RTP applies. However, the main applications for RTCP in a multi-party conference is to both control the traffic over one hop, as well as performing

conference media control [RFC5104], which means that e2e encryption cannot be applied in general. However, note that there are RTCP application messages, which might benefit from having e2e integrity protection.

2.2. SRTP Cloud Cryptographic Contexts

SRTP maintains a cryptographic context, containing master key(s), cryptographic transforms, etc., for the associated SRTP session. Exactly how the parameters in the cryptographic context are agreed upon is a session setup issue and out of scope of SRTP. SRTP assumes that a cryptographic context or rather the master key therein, is shared only between mutually trusted parties.

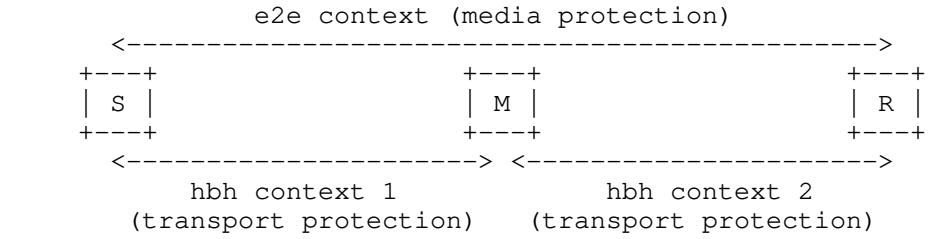


Figure 2: Context sharing (Sender, Middlebox, Receiver)

Note that in Figure 2, R represents multiple receivers in the case of multi-party conferencing. Also M may represent several cascading middleboxes.

The SRTP cryptographic context concept is reusable for the proposed solution. Conceptually, the originator and the intended end-receiver(s) share an e2e media security context, while a hbh transport security context is shared by an endpoint and an intermediary or by two intermediaries, see Figure 2.

The master key(s) in the e2e context MUST be cryptographically independent of, and MUST NOT be deducible from, the master key of any hbh context. The key management protocol(s) used MUST therefore be able to negotiate keys satisfying these requirements.

The identification of the hbh context is as defined in [RFC3711], while the used e2e context is implicitly identified in the session setup.

A sender will use two cryptographic contexts: an e2e context used for payload protection to the end-receiver(s), and a hbh context used to secure the SRTP transport to the (first) intermediary. Similarly, the end-receiver(s) will use two contexts. An intermediary node

however, will only use one standard SRTP context for each session. In other words, an e2e context is used to achieve transport independent media protection, and an hbh context is similarly used to achieve transport protection.

For both e2e and hbh contexts, it is assumed that cryptographic context parameters, such as master key and salt (if needed) are included. From these, session keys/salts are derived similarly to [RFC3711].

2.3. SRTP Cloud Packet Format

The packet format is composed of an "inner" e2e (sender-receiver) part embedded in an "outer" hbh (sender-middlebox or middlebox-receiver) part.

The SRTP Cloud packet format looks approximately like Figure 3

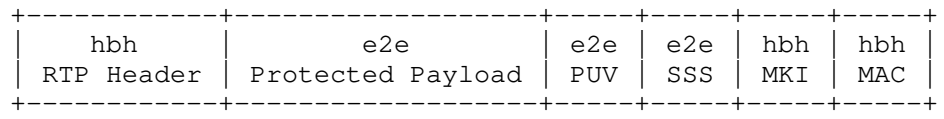


Figure 3: SRTP Cloud packet format

The additional fields added by the inner e2e security processing are:

- SSS: SRTP Source is a value used by the e2e transform as an identifier for the source within a e2e session. Thus, SSS MUST be unique for all sources within the session.

- PUV: Packet Unique Value for the e2e transform. The PUV shall be unique for each e2e protected payload being generated by a source within a e2e session.

The RTP header, hbh MAC, and hbh MKI are in one-to-one correspondence with respective fields of [RFC3711] and will not be discussed further.

2.4. Replay Protection

When the RTP data is hbh transport protected between server and receiver, replay protection on the transport level is provided as the hbh protection offers the same security features as [RFC3711]. It is assumed that the server is trusted not to attempt replay of data on media level, unless the user requests it and thus, this is in line with the trust model.

It is possible to implement replay protection on the media level for e2e transforms when the PUV is a counter. This has to be done on the application layer for the applications that requires it.

3. SRTP Cloud Specification

3.1. The Cloud Extension

The Cloud extension consists of a new packet format (Section 3.4), an extended cryptographic context concept (Section 3.5), and new SRTP processing at sender/receiver (Section 3.6). Considering only the cryptographic processing, cloud based middleboxes are compatible with [RFC3711], and the necessary additional processing is defined in Section 3.6.2. Senders/receivers need to support new cryptographic transforms (see Section 3.8).

3.2. Terminology

An e2e session is defined as the set of e2e protected data produced under a single e2e context (a security association between sender and the ultimate receiver(s), see Section 3.5.1 for the exact definition of e2e context). A e2e session may comprise several sources, i.e. several distinct logical e2e media streams to be protected by the same e2e context.

A hbh session is defined as the set of hbh protected data produced under a single hbh context (a security association between two entities, see Section 3.5 for the exact definition of hbh context).

The cryptographic transforms, keys, etc., used for the e2e and hbh protection, respectively, are denoted e2e transform, hbh transform, e2e key, hbh key, etc.

Throughout the specification all protocol data fields are assumed to be byte aligned, i.e. all defined bit-sizes SHALL be multiples of 8.

3.3. Trust Model

For the purpose of this document we use the following definitions:

A is said to trust B with information I, if A is willing to share I with B. In the sequel we will simply say that A trusts B.

A is said to have sender-semi-trust in B if A considers B to be "honest-but-curious" in the following sense. A trust B to maintain information I provided by A, and (later) redistribute it to the intended recipients as specified by A (parties that A trust with I). However, A does not trust that B will not also try to extract the

information I for him/herself and/or to attempt to distribute I also to other parties, e.g. parties that A does not trust with I.

A is similarly said to have receiver-semi-trust in B, if A trusts B to maintain information intended for A and to (later) distribute this information to A if and only if A so requests. However, A does not trust that B will not also attempt to distribute the information to other parties and/or try to extract it him/herself.

When it is obvious from the context (or irrelevant) we shall omit the directivity (sender/receiver) and simply say that A semi-trusts B.

Figure 4 shows the assumed trust model in terms of previous definitions.

In practice, the model means that

- S trusts R,
- S semi-trusts M to deliver information to R, and,
- R semi-trusts M to forward any information intended for R.

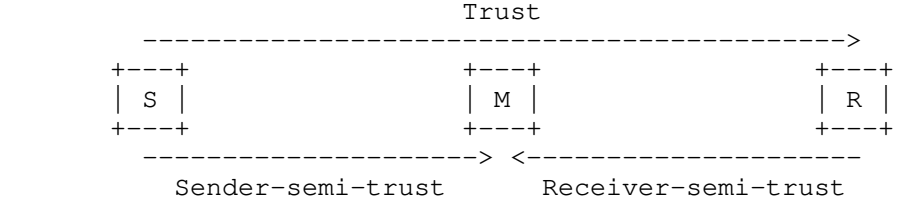


Figure 4: Trust Model (Sender, Middlebox, Receiver)

Note in the case of multi-party conferencing R represents multiple receivers.

As noted in the use cases above, S may be more concerned with who gets access to the information than R is. Still, this trust model, assuming a bare minimum of sender- and receiver-semi-trust as defined above, has been chosen since it is a simple trust model and seems to apply (qualitatively) as a common denominator for all the use cases where a cloud based middlebox is involved. Note also that the trust between S and R may often be mutual, but we do not require this.

M does not need to trust either of S or R. However, the trust model also assumes the existence of other parties (not shown) that are not trusted by any of S, M, or R, and which may attempt to intervene with the communication between them and the services provided by M. Thus,

depending on the application and the security requirements, authentication of S and R may be needed by some middleboxes in order to detect impersonation and prevent unauthorized access.

When there are several middleboxes in the path between S and R, it is necessary to assume that the middleboxes semi-trust each other, at least in a transitive sense. Also, we may then have a situation where S and R does not (directly) semi-trust a common M.

3.4. SRTP Cloud Packet Format

Figure 5 illustrates the format of the SRTP packet with the Cloud extension.

The packet format is composed of an "inner" e2e (sender-receiver) part embedded in an "outer" hbh (sender-middlebox or middlebox-receiver) part.

The e2e protected portion provides e2e encryption and authentication of the payload, RTP padding, and RTP pad count. The e2e protected portion also defines two new fields (PUV and SSS) for cryptographic synchronization. These two fields, together with the padding flag P in the RTP header, are e2e authenticated.

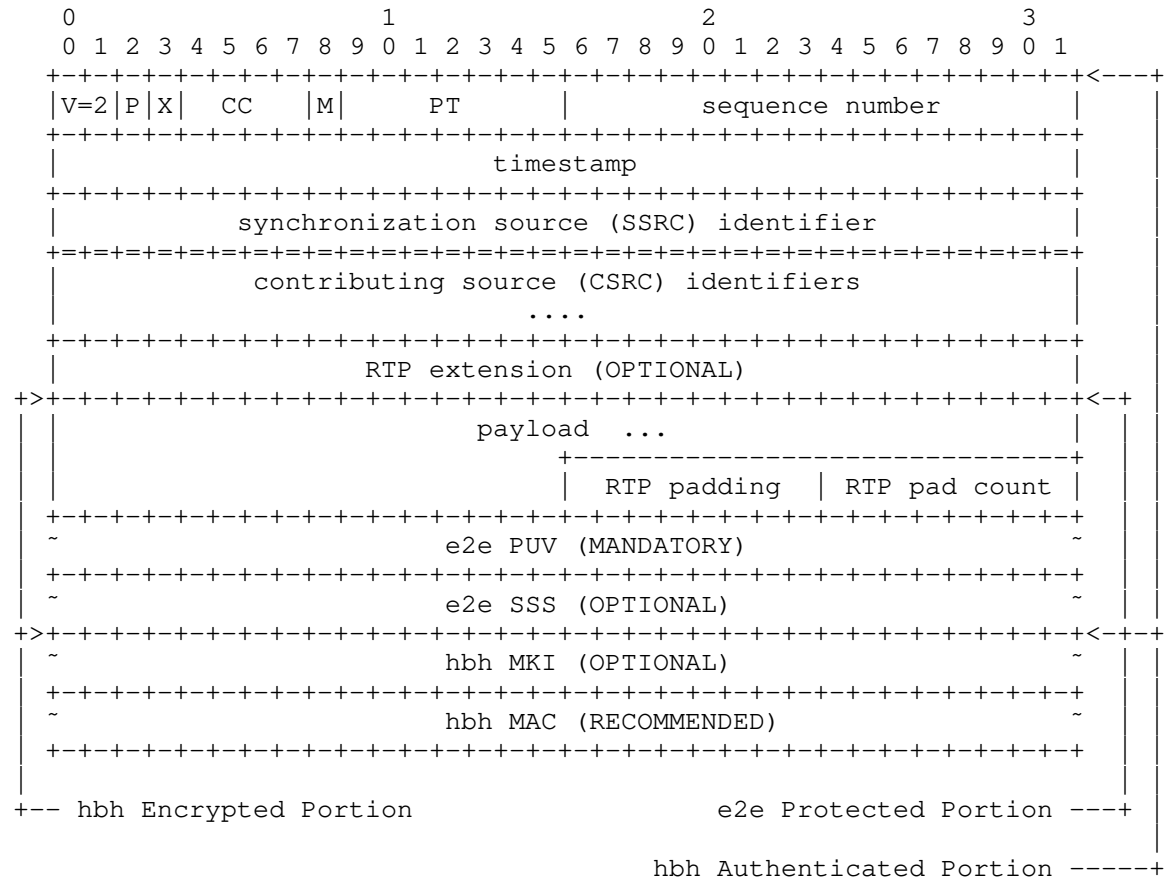


Figure 5: The format of the SRTP Cloud packet.

Default e2e transforms, which provide both encryption and authentication, and which SHALL be supported are defined in Section 3.8.3.

The e2e protected portion is opaque from the cloud based middlebox point-of-view.

Thus, by treating the inner e2e protected portion as the (hbh) "encrypted portion" of [RFC3711], the overall SRTP Cloud packet format conforms to standard [RFC3711] compliant SRTP. (Note that the additional fields added in the inner e2e part could just as well have been added by a new transform defined for SRTP, e.g. padding and/or crypto synch fields.) Hence, the hbh MAC and hbh MKI are in one-to-one correspondence with the MAC and MKI of [RFC3711] and will not be discussed further.

The additional fields added by the inner e2e security processing are:

- SSS: SRTP Source is a value used by the SRTP Cloud transform as an identifier for the source within an e2e session. Thus, SSS MUST be unique for all sources within the e2e session. Since there may be only one such source, the SSS field is OPTIONAL and of configurable length. SSS resembles the SSRC usage in RTP/SRTP in the sense that it ensures that two-time pads do not occur under the same e2e master key, see Sections 3.8 and 4.2. The implementation of the necessary anti-collision mechanism is outside the scope of this specification. The format is implementation specific, but the values 0, 1, 2, ..., n-1 MAY be used if there are n sources.

- PUV: Packet Unique Value for the e2e transform. PUV is transform dependent, of configurable length, and MANDATORY. The format is transform dependent and security aspects need to be considered when defining the format, see Sections 4.2 and 4.4. For a given e2e session and source, the PUV SHALL be unique for each generated e2e protected portion. The PUV is used as input to the IV formation for the e2e authenticated encryption transform.

Parameters which are configurable have default values (see Section 3.8.4), and are otherwise negotiated during e2e/hbh session establishment, agreed upon out of band, or hard coded for a specific application. The new fields are of configurable length for maximum data compactness (see Table 3.1).

Field	SRTP Counterpart	Typical Size (bytes)
PUV	SRTP Index	3
SSS	SSRC (IV formation)	0-1
Total		3-4

Table 3.1: Additional parameters

3.5. Extension of the SRTP Cryptographic Context

A SRTP Cloud cryptographic context SHALL consist of two main parts.

1. A hbh context. The hbh context SHALL be an SRTP cryptographic context conforming to [RFC3711] and SHALL be used for the hbh protection between sender and middlebox, between middlebox and receiver, or, between two middleboxes. The hbh context SHALL thus be identified by the <SSRC, destination network address, destination port number> triplet exactly as defined in [RFC3711].

2. An e2e context: this part of the context is defined below and SHALL be used for the e2e protection between sender and receiver(s).

3.5.1. Definition of e2e Context

The e2e context SHALL contain the following e2e transform independent parameters.

- an identifier for the e2e authenticated encryption algorithm, i.e., the AEAD cipher, see Section 3.8.3 for the default e2e transform specification,
- an identifier for the e2e pseudo-random function,
- an e2e master key, which MUST be random and secret to all except sender and receiver(s). The e2e master key MUST be cryptographically independent of any hbh key,
- an e2e master salt. Use of e2e master salt is strongly RECOMMENDED. This value, when used, MUST be random, but MAY be public.
- non-negative integers n_e, and n_s determining the length of the e2e session key for authenticated encryption and the e2e session salt.
- non-negative integers n_PUV, and n_SSS determining the length of the PUV, and SSS fields, respectively.

There may also be need to include e2e transform dependent parameters, see Section 3.8.4 for the parameters associated with the default e2e transforms.

Observe that there is no replay protection data in the e2e context, see Section 3.6.3.1. Also note that unlike [RFC3711] cryptographic contexts, the e2e context SHALL only contain parameters for RTP protection, and SHALL NOT contain parameters for RTCP protection, see Section 3.7.

Only end-points need to support e2e contexts, i.e. senders and receivers.

3.5.2. Identification of e2e Context

The e2e context SHALL be identified by out-of-band (outside the SRTP Cloud Packet) and in-band (in the SRTP Cloud Packet) signaling.

3.5.2.1. Out-of-band Signaling

The e2e context MAY be identified by simply transferring the entire context out-of-band (e.g. in the SIP signaling). Only half-roundtrip key management protocols can be used and the e2e context MUST be e2e protected so that middleboxes or other unauthorized entities cannot access or modify it.

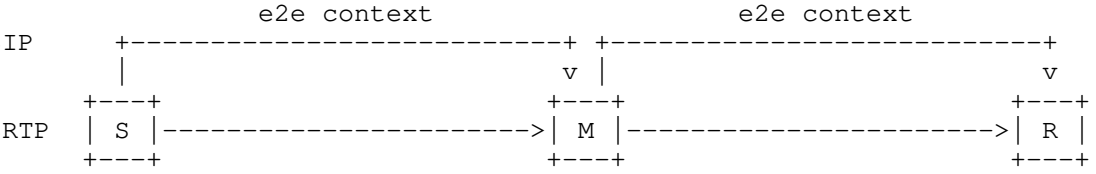


Figure 6: Transferring the entire e2e context via the middlebox

3.5.2.2. In-band Signaling

The in-band context identification is similar to that of [RFC3711] and SHALL be defined as follows. The e2e context is uniquely identified by the triplet context identifier:

<SSS, destination network address, destination port number>

The e2e context is here implicit from the triplet. Just like in [RFC3711] the entire triplet MAY not be needed to uniquely identify the e2e context. In the case of multi-party conferencing where there are multiple receivers, SSS alone identifies the e2e context.

3.6. SRTP Cloud Processing

In what follows, it is assumed that the sender and receiver(s) agree out-of-band on the e2e cryptographic context.

It is similarly assumed that sender-middlebox and middlebox-receiver, respectively, agree on the hbh cryptographic context.

3.6.1. Sender

The sender SHALL first, out-of-band, establish the necessary hbh context parameters with the middlebox as discussed above. The rest of sender's processing is identical to [RFC3711] with the following exceptions and extensions.

S1 In analogy with step 1 of [RFC3711], the sender SHALL determine both the hbh context and the e2e context as discussed in

Section 3.5. Next, and prior to performing step 2 of [RFC3711], the sender SHALL perform step S2-S4 as defined below.

S2 The sender SHALL from the e2e master key and master salt derive the e2e session key/salt as described in Section 3.8.2.

S3 The sender SHALL next apply the e2e transform as described in Section 3.8.3.

S4 The sender SHALL then form the e2e protected portion of the SRTP Cloud packet by concatenating the result of S3, the PUV, and the SSS.

The rest of the sender's processing conforms to [RFC3711], steps 2-8, by treating the result of S5 as the part to be encrypted ("encrypted portion" of [RFC3711]) and using the hbh context.

3.6.2. Middlebox

Middleboxes do not have access to the e2e contexts and may even be unaware of their definition. Hence, "context" in this section refers to standard [RFC3711] cryptographic contexts, which in turn agrees with the hbh contexts defined herein.

Generally, the middlebox SHALL first, out-of-band, establish the necessary hbh context parameters with the source or destination.

3.6.2.1. Acting as Receiver

MR1 When receiving media from a sender, the middlebox SHALL retrieve the correct context and process the packet exactly according to the receiver behavior of [RFC3711].

MR2 The middlebox SHALL store sufficient information to later be able to map the correct content to the intended receiver, e.g. e2e context, or the intended receiver's identity (ID). ID format and usage is otherwise out of scope for this specification, but could, e.g., be retrieved during the session establishment.

MR3 The middlebox SHALL store information sufficient to later reconstruct the e2e protected portion of the packets (corresponding to Figure 5) and to allow the receiver to uniquely identify the correct e2e context, e.g. by storing the the e2e context. Note that header information (e.g. P, M, PT, and timestamp) is needed for rendering and information from RTCP SR is used for synchronization between streams e.g. in a

multimedia video/audio session. Such information also has to be stored by the middlebox.

3.6.2.2. Acting as Sender

MS1 When forwarding media to the receiver, the middlebox SHALL retrieve the correct hbh context as specified in [RFC3711].

MS2 The e2e protected portion SHALL be used as the payload.

MS3 An RTP Header SHALL be formed with the P and M fields identical to when the payload was stored. The PT field must refer to a PT definition that is equivalent to the one received. The original timestamp MAY be used. The SEQ and SSRC SHOULD be defined strictly on hbh basis.

MS4 The middlebox SHALL then concatenate the payload from MS2 with the RTP header from MS3 and process the packet exactly according to the sender behavior of [RFC3711] using the retrieved context. As noted above, certain information from RTCP messages, originating from the sender (e.g. RTCP SRs), may also need to be forwarded (and sometimes modified as discussed in Section 4.6). These (and other RTCP messages) SHALL be processed according to the SRTCP specification of [RFC3711].

3.6.2.3. Multiple Middleboxes

When more than one middlebox is present, we consider a pair of adjacent middleboxes M1 and M2, where M1 forwards media to M2.

M1 SHALL act as a middlebox sender (Section 3.6.2.2) treating M2 as a receiver. M2 SHALL act as a middlebox receiver (Section 3.6.2.1) treating M1 as a sender.

3.6.3. Receiver

R1 The receiver SHALL first, out-of-band, establish the necessary hbh context parameters with the middlebox.

R2 Step 1 to 8 of [RFC3711] SHALL then be applied, using the hbh context to perform hbh processing.

The remainder of the processing concerns the e2e protection. The result after performing the hbh authentication check and decryption as described above MAY be stored at the receiver for later application of the e2e processing. If so, the receiver MUST store the e2e protected portion in order to be able to perform the further steps as described below.

R3 The receiver SHALL next determine the e2e context as discussed in Section 3.5.2.

R4 The receiver SHALL derive the e2e session encryption key as described in Section 3.8.2 using the e2e master key and salt.

R5 The receiver SHALL verify authentication and decrypt the e2e protected portion as specified by the e2e transform(s), see Section 3.8.3. If the result of authentication is "FAILURE", the packet MUST be discarded from further processing and the event SHOULD be logged. Note that there is no replay protection for the e2e context (see Section 4.4).

R6 The receiver removes PUV, and SSS as appropriate.

3.6.3.1. Replay Protection

For reasons discussed in Appendix A, it is in general not meaningful or desirable to provide application independent replay protection for the e2e part. Some of the identified use cases make this clear by having a requirement that the receiver should be able to jump back/ forward in the e2e media stream. See Section 4.4 for security considerations.

3.7. Use of SRTCP with SRTP Cloud

SRTCP protection SHALL be provided hbh, conforming to [RFC3711], and SHALL NOT be provided e2e, as this covers most/all use cases currently identified. Protecting e.g. the synchronization information e2e would prevent use cases where several stored streams are spliced together. Further RFCs may specify additional e2e functionality for SRTCP Cloud.

As noted, it may still be needed to forward information from some of the inbound RTCP messages (e.g. RTCP SR and APP). Note that if several stored streams are spliced together, the timestamps and therefore also the synchronization information has to be modified. Also note that it may in general not be possible for the middlebox to reproduce RTCP reports accurately reflecting the ongoing hbh session. For instance, since the e2e encryption hides any possible RTP padding, there may be a discrepancy between sender's byte counts on the S-M and M-R links, respectively. After decryption at R, however, the correct values will be possible to reconstruct.

3.8. Cryptographic Transforms

We define a set of pre-defined SRTP Cloud e2e transforms. Note that middleboxes do not need to support any cryptographic transform outside what is already defined in [RFC3711]. The hbh protection may reuse any of the existing SRTP transforms such as those defined in the original specification [RFC3711], or, transforms that have been added later. The e2e protection may use the transforms defined in Section 3.8.1, or, transforms that have been added later (see Section 3.8.5).

3.8.1. Pre-Defined e2e Transforms

For e2e protection we use Authenticated Encryption with Associated Data (AEAD) algorithms. Specifically, the AES-GCM (AES Galois/ Counter Mode) transforms as specified in [I-D.ietf-avtcore-srtp-aes-gcm] are used as the pre-defined e2e cryptographic transforms, with the following modifications.

Instead of forming the initialization vector as specified in Section 9.1 of [I-D.ietf-avtcore-srtp-aes-gcm], the IV SHALL be formed by first concatenating 2-octets of zeroes, a 4-octet SSS (the original SSS padded with as many leading zeros as needed) and a 6-octet PUV (the original PUV padded with as many leading zeros as needed). The resulting 12-octet value is then bitwise-XORed to the 12-octet session salt to form the 12-octet IV.

SSS and PUV are the SSS/PUV fields from the packet. The PUV is a counter, initially set to zero and then increasing by one (1) for each packet. The maximum allowed size of the PUV for AES-GCM SHALL be 48 bits. If the SSS field is not present, the value 0 (zero) SHALL be used. The maximum allowed size of the SSS for AES-GCM SHALL be 32 bits.

The associated data specified in Section 9.2 of [I-D.ietf-avtcore-srtp-aes-gcm] is redefined as:

Associated Data: The padding flag P (1 bit), PUV (n_PUV bits) and SSS (if used, n_SSS bits)

3.8.2. Session Key Derivation

For the hbh security processing, session key derivation SHALL be done exactly as in [RFC3711] using the hbh master key and salt.

For the e2e security processing the key derivation is also identical to [RFC3711] with the following exceptions

- The e2e master key and salt, SHALL be used together with the defined labels of [RFC3711] for derivation of the different keys.

- The key derivation rate SHALL be zero.

3.8.3. Default Transforms

The default hbh encryption transform SHALL be the NULL encryption algorithm, the default hbh authentication transform SHALL be HMAC-SHA1, and the default hbh pseudo-random function SHALL be AES-CM. The transforms are defined in Sections 4.1.1, 4.2.1, and 4.3.3 of [RFC3711].

The default e2e cryptographic transforms SHALL be AES-GCM as defined in Section 3.8.1. The default e2e pseudo-random function SHALL be AES-CM as defined in [RFC3711], Section 4.3.3.

3.8.4. SRTP Cloud Default Parameters

The default hbh parameters SHALL be identical to [RFC3711].

The default e2e parameters for master and session key lengths are the same as in [RFC3711] with the differences in transform definition as defined above and the following additional exception.

- Replay window size: N/A (or 0).

We also add the following additional bit-length parameters:

parameter	min	default
n_PUV	16	24
n_SSS	0	0

Table 3.2: Additional parameters

3.8.5. Adding Future e2e Transforms

Adding transforms for the hbh protection SHALL follow the existing guidelines of [RFC3711]. Indeed, any current (or future, as far as we can see) transform specification for SRTP is applicable for usage with the hbh protection.

To add an e2e transform, the accompanying specification MUST, besides specifying the cryptographic operations, define the format and usage of the PUV field and, if used, for the SSS field and any possible additional field, e.g. padding.

4. Security Considerations

4.1. General

Though it may seem that there are quite a few differences between the cryptography and key management used in [RFC3711] and the corresponding functions defined here, the differences are actually smaller than one may think and the security considerations turn out to be essentially equivalent.

As noted, a problem of SRTP in applications with cloud based middlebox is the transforms' dependence of the SSRC. The SSRC is part of IV formation and crypto context identification in [RFC3711].

In this specification two new in-band parameters, PUV and SSS, are specified. Note that SSS is used in exactly the same way the SSRC is used in [RFC3711]: IV formation. Basically, one can think of the SSS as the e2e source identifier. The SSS is e2e protected.

4.2. Keystream Reuse

A main concern of [RFC3711] is to avoid keystream reuse. This concern is present also here. The currently defined encryption transforms are additive stream ciphers, which are sensitive to keystream reuse. It is therefore RECOMMENDED that each session utilizes random and cryptographically independent e2e and hbh keys.

When sender and receiver share an e2e master key it may be convenient to reuse the key for several e2e sessions/messages via the middlebox. Another situation when key reuse may be beneficial is if sender and receiver use the middlebox in a "chat-like" fashion (with bi-directional communication using the same e2e master key in both directions). In this case there may be a risk that a message in one direction (e.g. "A-to-B") reuses keystream of some message in the other direction ("B-to-A"). For the predefined e2e encryption transform such reuse will only be secure if the sender and receiver keep state to prohibit reuse of IVs.

Unique IVs MAY be assured by putting requirement on the implementation of the sender to ensure that unique SSS values are used each time the same e2e master key is reused. For the bidirectional case (as well as for the more general case where a group key is used as e2e master key), some out-of-band signaling that assures that end-points use distinct SSSs is, as mentioned, REQUIRED.

The situation is essentially equivalent to that of SRTP. As noted in the security considerations of [RFC3711], keys may be reused (with the predefined transforms) if (and only if) unique SSRC values can be

guaranteed. Due to the risks of misuse, reuse of master keys between sessions is, just as in [RFC3711], therefore NOT RECOMMENDED.

4.3. Authentication and Authorization

For reasons already discussed, it is RECOMMENDED that middleboxes authorize senders and receivers (typically involving authentication) before accepting/forwarding messages. While the content is protected by keys supposedly only known to the receiver(s), this provides extra protection if the e2e keys have fallen into the wrong hands and it also avoids that the middlebox wastes resources, responding to spoofed requests. It is also RECOMMENDED to have e2e authentication between sender and receiver(s), which is achieved by applying authentication/integrity to the e2e protected portion.

4.4. Replay Protection

Replay protection is provided on an hbh basis by use of an hbh transform including message authentication. It is RECOMMENDED to use hbh message authentication as it protects from outsiders attempting to change the order of packets.

Since some scenarios considered makes it reasonable to expect that the receiver may wish to jump (fast-forward or rewind) in the e2e protected media flow, it is not meaningful to strictly enforce replay protection on an e2e basis. Note however that our trust model assumes that the middleboxes are trusted enough not to attempt to replay or reorder media unless the receiver so requests.

It is however still possible (and RECOMMENDED) to provide e2e authentication of the packets in combination with inclusion of a sequence number in the PUV (as the default e2e transform does). It then becomes infeasible even for the middlebox to fake the relative association between a particular packet and its sequence number. This means that the receiver will be able to detect a replay that occurs without the receiver actually having requested it.

4.5. Key Management Considerations

Key management is outside the scope of this specification which is an intentional design choice in order not to introduce any dependency on using a specific key management scheme. Nevertheless, some considerations need to be highlighted and taken into account when deploying this specification in practice.

To implement the targeted trust model, the main concern is that the e2e keys MUST be independent from the hbh keys. In other words

knowledge of any hbh key MUST NOT reveal non-trivial information about any e2e key.

This can be achieved by ensuring that key management for hbh and e2e protection is carried out independently using fresh, random and independent keys each time. This is the RECOMMENDED approach.

Another alternative which may be attractive in some cases is to use the slightly weaker notion of cryptographic independence. Here, the hbh keys MAY be derived from the e2e keys by applying a sufficiently strong pseudo-random function.

Even if hbh keys are random and independent each time, it is still RECOMMENDED that e2e keys are not cached/reused (see Section 4.2 for discussion on keystream reuse).

4.6. Privacy

In order for a cloud based middlebox to deliver the correct media (produced with the correct e2e context) to the receiver(s), some applications may choose to store information regarding the identity of the sender and will be able to deduce the communication taking part between the two.

To enhance privacy, senders/receivers may use agreed pseudonyms or other similar Privacy Enhancing Techniques (PET)s. Complete anonymity may be in conflict with the requirement that the middlebox needs protection from flooding by garbage or other forms of unwanted traffic.

4.7. RTCP Considerations

As specified, RTCP is only protected on hbh basis. This is motivated by the assumption that a middlebox indeed is a true store-and-forward entity (as opposed to performing a more intelligent function). The inbound/outbound RTP sessions are then different and RTCP then reports only on the current RTP session. As noted though, it may still be useful to forward e.g. (modified) sender reports to the receiver using hbh RTCP protection.

4.8. Malicious middleboxes

Middleboxes are semi-trusted which implies that they are assumed to (at least) forward data as requested by the sender/receiver. Malicious middleboxes therefore falls outside the trust model. Nevertheless, even if a middlebox is malicious beyond our assumptions, such attacks will only have DoS effects if e2e authentication is used (RECOMMENDED) and could as easily have been

done by some other party (non-middlebox). If hbh authentication is used (RECOMMENDED), it can even be detected that it is the middlebox that modified the e2e protected part.

By modifying RTCP, a malicious middleboxes could perform attacks that are not detected but which would be detected if done by some other party (non-middlebox). By incorrectly altering RTCP SR packets, a malicious middlebox could forward only parts of messages or mess with the synchronization information without being detected. However, for the intended use cases, there seems to be no gain to the middlebox owner (typically a cloud service provider or network operator) to perform such attacks to its paying customers.

5. Contributors

Yi Cheng has contributed to this document and was an author of previous versions. She has been moved to the Contributors section as her affiliation and contact address are unknown.

6. Acknowledgements

The authors would like to give special thanks to Magnus Westerlund for his valuable comments and feedback.

7. IANA Considerations

To signal that the new transforms are used, each relevant key management protocol needs to register the new transforms including numbering scheme and syntax with IANA.

8. References

8.1. Normative References

[I-D.ietf-avtcore-srtp-aes-gcm]
McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in Secure RTP (SRTP)", draft-ietf-avtcore-srtp-aes-gcm-17 (work in progress), June 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

8.2. Informative References

[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, February 2006.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

Appendix A. Use Cases

A.1. Problem Statement

We consider RTP communication solutions that include semi-trusted middleboxes, i.e. middleboxes that should not have access to cleartext media, but still should be able to have access to other data in order to retransmit media according to RTP standard procedures. Below, we provide some use cases where S, M, and R refer to Sender, Middlebox, and Receiver. For each use case, we comment on aspects of the trust-model defined above.

Cloud based conferencing: A conference participant (S) talks to other participants (R) via a conference bridge (M) that is hosted in the cloud. Since the participants have no control over the cloud environment which might be under pervasive monitoring, S and R do not want the content of their communications to be disclosed to M. M is only trusted to be a forwarder of (encrypted) media.

Caching Protected Media in the Network: A content provider (S) broadcasts high value encrypted media (e.g. Internet Protocol Television (IPTV), and radio) to clients (R). A network node (M) is enhancing distribution by caching of the media, but is not trusted by the content provider and has therefore no access to the encryption keys. A client that missed the beginning of a program might stream the media from the network cache instead of listening to the broadcast. Due to the trust model where the content provider only trusts the clients, the media needs to be e2e protected. Nevertheless, the media also needs to be hbh integrity protected to protect against denial-of-service (DoS) attacks.

The typical use case is thus to require that media is (at least) confidentiality protected end-to-end (e2e) between the sender and the receiver. At the same time the communication should be protected hop-by-hop (hbh) to prevent malicious users from performing denial of service attacks by sending bogus data to middleboxes or replaying previous packets.

A.2. Security Requirements

The security requirements for SRTP Cloud are:

- Transport independent media protection

  It SHALL be possible to have media protection that is independent of RTP parameters.

  To allow retransmission of received protected media, a transform for protecting the RTP payload that is independent of RTP transport parameters is needed.

  The media protection MUST cover both message authentication and confidentiality protection.

  It SHALL be possible to protect several e2e protected media streams with a single e2e context.

- Media source authentication

  It SHALL be possible to provide e2e source authentication of the media stream.

  In a group setting, source authentication is here meant to ensure that the message originated from a member of the group. This requirement is fulfilled if media has authentication protection in a transport independent manner.

- Support of playback of protected media streams

  A client SHALL be able to do random seek in a protected media stream.

  Note that as playback functions like retransmission and random seek capability are features in the described use cases, replay protection cannot be required for transport independent media protection.

- Transport protection

  It SHALL be possible to provide transport protection that is independent of the media protection.

  The transport protection MUST be able to provide confidentiality, authentication, and replay protection for RTP and at least authentication and replay protection for RTCP.

This requirement maps well against SRTP as of [RFC3711]. Transport protection is also a means to provide replay protection of the media on a hop-by-hop basis.

- o Separation of security contexts

  It MUST be possible to have independent security contexts for the transport independent media protection and the transport protection.

  This means in particular that there has to be two distinct master keys, one for e2e media protection and one for hbh transport protection.

A.3. Problems with SRTP in Cloud Based Scenarios

It would be desirable to be able to offer use of SRTP as a general, lightweight mechanism to achieve the above type of protection, but trying to do so reveals two main problems.

The first problem is due to the fact that RTP streams received and later resent by an middlebox in general are independent; received SRTP-encrypted payloads cannot just be retransmitted as a new SSRC is most likely used when retransmitting. And if several recorded streams are spliced together, an offset must be added to the SEQ and timestamp so that they form a continuous sequence. This in particular implies that SRTP with currently defined transforms cannot be applied end-to-end as they depend on the RTP header.

The second problem is that in order to provide both e2e and hbh protection, two independent security contexts with associated protection mechanisms have to coexist; a feature unavailable in SRTP as currently specified. While it is not too difficult to imagine how two contexts in place of one might be used, a problem arises when specifying how the e2e part of the context should be identified and signaled, as current SRTP context definition rests on parameters which are not constant end-to-end in the scenario where a middlebox is involved, namely SSRC and receiver's IP address and port.

The SRTP extensions defined in this document address these problems.

A.4. Design Rationale

As noted above, different use cases may have slightly different security requirements and trust levels and there may be many different possibilities to extend SRTP in different directions to handle a specific use case (or some subset of use cases). For

example, the problems related to the most basic trust model extension (need to provide confidentiality e2e and integrity hbh) are due to the fact that in SRTP, parties always know both the encryption key and the authentication key. This could be addressed (mainly) by just separating encryption and authentication keys (i.e. modifying SRTP key derivation and cryptographic context). However, the solution would then become severely limited, e.g. it would not support pre-encryption of data or re-transmission of stored data. Similarly, as will be seen below, SRTP Cloud adds some additional in-band data fields, though some use cases above could probably be handled without them. Again, the solution would be limited to these use-cases and would then not allow e.g. the secure fast-forward/rewind use cases, which requires in-band synchronization data. By making the added fields optional, it is possible to support these features as needed, yet keeping bandwidth low when such features are not needed.

Another approach would be to use some already defined standard like S/MIME or OpenPGP, which are mostly used for secure email. This works well when the entire message is e2e protected and transferred as a file from sender to middlebox and from middlebox to receiver, but it does not support streaming. Another drawback is that both S/MIME and OpenPGP require the use of public keys. Protecting each RTP packet with both SRTP, and S/MIME or OpenPGP would support streaming but would add significant overhead. Use of (D)TLS or IPsec is clearly ruled out since it would only provide hbh protection.

This specification is rather based on

- identifying the common denominator(s) to the cloud based middlebox problems, captured in the trust model and requirements of Section 3.2
- proposing a single extension of the SRTP framework (see Section 4.1) powerful enough to handle all foreseen middlebox use cases, which, by
- simple configuration of the extended framework (Section 4.3) can be adopted to support the requirements of the specific use case at hand.

Considering that the impacts of the present specification on SRTP are very similar to those of [RFC4383], there does not appear to be any disadvantage in having a single extension compared to having per-use-case extensions.

Authors' Addresses

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

Mats Naslund
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 33 739
Email: mats.naslund@ericsson.com

PERC
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

J. Mattsson
M. Naslund
M. Westerlund
Ericsson

October 19, 2015

Secure Real-time Transport Protocol (SRTP) for Cloud Services
draft-mattsson-perc-srtp-cloud-01

Abstract

In order to support use cases when two or more end-points communicate via one (or more) cloud service (e.g. virtualized cloud-based conferencing) that are not trusted to access the media content, this document describes the use of so called end-to-end (inner) and hop-by-hop (outer) cryptographic transforms within the Secure Real-time Transport Protocol (SRTP). One of the main aspects of the transforms is to make the confidentiality and message authentication independent of the RTP header. Another central aspect is to enable identification of the cryptographic contexts (keys etc.). Besides the security of the end-points, also trust assumptions regarding the cloud services are addressed.

Table of Contents

1. Introduction

This document proposes a solution to achieve End-to-End Security for an RTP media streams and its meta data within the context of PERC. However, the document focuses on the RTP e2e protection mechanism. It only puts requirements on the key-management, not proposing a solution for that part. This draft is a complete rewrite, and the proposal is based on what was sent to the PERC mailing list, but substantially updated based on feedback and discussion.

The discussion in this document is be based on the analysis of the RTP fields and how they needs to be handled written up in [I-D.westerlund-perc-rtp-field-considerations]. We will assume that the reader is familiar with that discussion.

This document assumes a basic model for protection of the data that consists of the following high level functions. A end-to-end media data protection mechanism as defined below in Section 3. An inband key-distribution mechanism to provide endpoints with RTP stream specific keys, assumed to be EKT based, whose requirements are discussed in Section 5. An key-management function that provides authorized endpoints with a EKT master key (Group key) (Section 6). In addition an hop-by-hop security mechanism (Section 4) are in use to protect that not possible to cover end-to-end with protection as necessary between the endpoints and middlboxes (MDD) in the system.

2. Definitions

This section provides a set of definitions.

2.1. Terminology

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 RFC 2119 [RFC2119].

This document uses the following terms:

Endpoint: An RTP stream sending and/or receiving entity that is part of the end-to-end security context.

MDD: Media Delivery Device - An RTP middlebox that operates according to any of the three possible RTP topologies [I-D.ietf-avtcore-rtp-topologies-update] that is possible in the PERC system:

Transport Translator - Relay

Switching RTP Mixer

Selective Forwarding Middlebox (SFM)

Third Party: An entity that is neither an endpoint nor an MDD.

3. End-To-End Security

This section discusses the various components of the end-to-end security for the media data, the RTP header fields, RTP header extensions, and media related RTCP messages and information.

3.1. End-to-End Payload

The end-to-end payload consist of a couple of destinct parts that all needs to be considered:

Media Payload: The Media Payload containing the information packed according to the RTP payload format in use.

Padding: 0 to 256 bytes of padding octets. Used to obfuscate the RTP payload length when necessary.

RTP Header Fields: A number of RTP header fields are closely associated with the media payload. They will be discussed below.

The RTP header fields can be classified into several different categoriezes:

E2E related, non-changable by MDD:

P

M

Header Extensions

E2E related, changable by MDD:

PT

Sequence number:

Timestamp:

SSRC/CSRC: Identifying the RTP Stream

Header Extensions:

Hop-by-hop:

V:

X:

CC:

Header Extensions:

As can be seen there are some fields that can be closely tied with the payload and which MUST be forwarded by the MDD unaltered. We have other fields that needs to be possible to change by the MDD. For some of these it is critical that a receiving endpoint can learn the original value to verify that the MDD's actions as acceptable. Then there are some fields that are fixed in the protocol like V or otherwise needs to be handled on a hop-by-hop basis. For the X and CC field their value is dependent on the need to carry some field in their corresponding data fields the Extension header and CSRC list respectively.

The most problematic fields are the ones that have to be rewritten, but still have important indicator purposes, like PT, SSRC/CSRC, Timestamp and Header Extension IDs. Sequence number is not included, just because we know it is so critical to know the relative order of transmitted packets that this MUST be preserved end-to-end. Original Timestamp we will discuss below in Replay Protection (Section 3.3).

Our proposal for the end-to-end media payload is the following:

The SSRC is assigned uniquely by a higher management function. For media switching mixers, that original value is preserved using the CSRC field. Any MDD that receives a RTP stream that is a switched one, i.e. the SSRC belongs to the MDD, rather than originating endpoint, and thus contain an CSRC field, will have to copy forward the CSRC value, not the MDD's SSRC value into the produced outgoing packet's CSRC field.

> Note: We can support MDD that makes SSRC/CSRC translations, but for that to work we strongly recommend to mirror the original SSRC into the inband key-management protocol. This to ensure that the unqiue stream identifier is preserved and can be verified and tied to other functions and verifications.

The SSRC/CSRC field will be used by the receiving endpoint as a reference to the security context established by the combination of the RTP packets received and the information from the inband key-management protocol. Any on-path SSRC/CSRC translation will be possible as the receiving endpoint will only use the received value as a reference to the context, not part of the protection operation. The important is that the originating SSRC is consistently handled by the system.

Each sent RTP packet from the originating endpoint will have in place of the regular RTP payload an security protected end-to-end payload. This payload will consist of 32-bit of end-to-end sequence number followed by a variable number of bytes of security payload. The security payload is created by applying the cipher (AES-GCM [I-D.ietf-avtcore-srtp-aes-gcm] assumed) with as plaintext: RTP payload format, Padding, and as associated data: P, M, PT (Original), Timestamp (Original), End-to-end header extensions (both confidentiality and only authenticated one in plain text).

As Initialization vector (IV) to the cipher the following data is used: HeaderExtFlag (1 bit) : NullFlags(7 bits) : NullPadding (24 bits) : Stream Specific Key Sequence Nr (32 bits) : End-to-End Sequence number (32-bit). These values are first concatenated and then XORed with the e2e session salt to form the e2e IV. The stream specific key sequence number combined with the E2E sequence number forms an always increasing value for this particular RTP stream as identified by the unique stream ID (Orignal SSRC). The HeaderExtFlag is set to 0 for protection of the end-to-end payload (this section) and set to 1 for the confidentiality and integrity operation of any RTP header Extensions (Section 3.2).

3.2. RTP Header Extensions

There exist RTP header extensions that needs to be end-to-end authenticated. For these we want to ensure two important properties. A MDD shall not be able to remove it without detecting the removal, secondly, the integrity of the content of the header extension must be verified. This is proposed by including the header extensions that are marked as requiring end-to-end authentication in the e2e associated data for the packet.

This both the advantage and downside of being closely tied to the payload of the packet. This is advantageous as it prevents the MDD from interfeering with the information and when it is provided. However, it prevents the MDD to include relevant meta data in header extensions at its own descretion. One use case for this is Source description information like CNAME and MID that can be included with a stream when a new endpoint joins the conference.

A complicating factor is that like the PT (payload type) the ID field for header extensions are dynamically assigned, and the mapping can be endpoint specific. Which requires that the MDD can translate them as needed. This makes it difficult for the receiving endpoint to verify which format the source truly indicated. Preferably one should have a mechanism to indicate the original ID, so that the original value can be included in the associated data.

For RTP header extensions that requires confidentiality each header extension's data part is individually encrypted using a stream cipher. AES-GCM is not recommended to use due to the expansion that the internal integrity tag. The key will be the one associated with the source stream ID crypto context. The IV needs to contain: HeaderExtFlag (1 bit = 1) : in packet order : Padding : Stream Specific Key Sequence Nr (32 bits) : End-to-End Sequence number (32-bit). The "in packet order" number is a counter for each header extension included in the packet. So the first header extension gets zero (0), the next one (1), and so on. This results in that an MDD MUST retain the individual order of the header extensions when forwarding them. We also note that by individually protecting each header extension, any header extension where the ID is the information, i.e. without any data there will be no confidentiality.

> Note: The reason to initially consider this strucutre is that is can avoid forcing a move to 2-byte header extension headers. If one defines a new header extension that wraps all the header extensions including their ID and Lengths then it is likely that this becomes longer than 15 bytes. It also locks in the IDs which forces the receiving endpoint to know how the IDs at the originating source maps to specific extensions.

The authentication process for header extension is performed by taking each header extension in the order it was received and concatenate them togehter with the ID and length values in the field form used by two-bytes header extensions, independently which form they where received in. This avoids authentication errors if an MDD needs to switch between one and two bytes header extension format. The ID field is replaced with a null value. Having the original IDs would be preferable, as it would like for the PT enable verification of the intended format. This block of data is included as Associated data in the decryption.

3.3. Replay Protection++

This section is called Replay Protection++ as it is not only replay protection that is needed. Yes, replay protection is needed against replay attacks (Section 7.2.2), but also protection against a delayed playout attack (Section 7.2.3). In addition the mechanism needs to be robust against splicing attacks (Section 7.2.4) where the attacker attempts to provide another stream as this source's one.

The protection mechanism against these attacks works as follows. First the receiver tracks the source ID associated with a crypto context. Every time a new key is provided by an EKT message the receiver needs to verify that the source ID matches with the one in

the context. Next the key sequence number must be larger than stored otherwise the key in the context is kept.

When an RTP packet is received the crypto context is retrieved. The context stores the highest end-to-end sequence number received, and a vector indicating which of the last N packets that has been received, this to accept re-ordered packets that is only slightly delayed. It also stores the reception time and corresponding original timestamp value. First it forms the extended e2e sequence number concatinating the key sequence number (from EKT message if included and verified, or from context) with the e2e sequence number. If that is greater than the stored highest seen extended sequence number or within the window of acceptable older packets and not previously received, then the processing continues.

Then the time based checks are performed. The reception time delta is compared with the RTP timestamp difference. That difference must be within a error of margin equal to network jitter boundaries and an allowed fudge factor for media switching mixers to align content when switching. The margin of error is no larger than one or two seconds. If the difference is bigger than this MUST be indicated to the user if played out, discarding media is recommended.

For this method to be robust clock drift between sender and receiver needs to be tracked, as the RTP timestamp is based on the originating endpoint's clock and the reception time uses the receivers clock. Clock drift is only expected to be a significant issue if there is very long periods when no RTP packets are received with media from a particular sender. Using RTCP SR the receiver can track sample clock versus senders general clock. Every time a timestamped packet is received, SR or RTP they can be used to estimate the relative clock speed difference between endpoints.

Next the decryption including authentication is performed. If the received data is validated, then the crypto context is updated with the new highest extended sequence number as well as the time parameters.

3.4. RTCP

There exist a need for both end-to-end authenticated RTCP messages, as well as end-to-end confidentiality protected ones. When it comes to confidentiality protected ones, these includes end-to-end codec control [RFC5104], such as region of interest [3GPP TS 26.114, version 13.1.0]. The ROI feedback message is used by a receiver to request to view a particular region of the total captured frame. There exist no reason for the MDD to know what region that is requested by which users. If some of the defined RTCP SDES items was

to be used, like name, phone, location, and tool, there is significant privacy concerns around those, and they should be transported such that the MDD can't get access to them. Other SDES items like CNAME, MID are meta data related to the session. They can be generated in such a way that there are no privacy concerns with them. However, one would like to ensure that they are integrity protected to prevent any modification on the path from the sender to any receiver.

There also appears to be need for end-to-end messages providing vital information about each end-points actions, that can't be modified by the MDD. This is to enable auditing of the MDDs and prevent that they attempt to fool the users of them. For each unique e2e stream id each receiver should know how much packets actually was transmitted, what the current timestamp value, and corresponding wall clock time, preferably in a time base that can be tracked.

Below we propose how to address these cases.

3.4.1. End-to-End Authenticated RTCP

The end-to-end authenticated RTCP is a new RTCP packet type used as authentication wrapper. The new RTCP wrapper packet has the RTCP basic header identifying the packet type, the originating SSRC, the original SSRC (Source ID), a sequence number and an transmission timestamp (NTP format) and includes one or more regular RTCP packets with the information that needs to be end-to-end authenticated. This may lead to that what before would have been multiple items in on RTCP packet, now becomes divided on multiple RTCP packets types based on its security classification. The whole packet with the exception of the originating SSRC field is authenticated (associated data), and the tag (AES-GCM output) located last in the wrapper RTCP packet.

3.4.2. End-to-End Confidential RTCP

Similar to the e2e authenticated RTCP this packet is also a wrapper for one or more RTCP packets that need to handled confidential end-to-end. The 64 byte common RTPC packet header is not encrypted. This is followed by a original SSRC (Source ID) field, a sequence number used to build the IV for the packet. The part that is confidentiality protected is the transmission timestamp, the RTCP packets (in fully), and any RTCP padding.

4. Hop-by-Hop Security

We have considered that two different Hop-by-Hop security protocols may be used between an end-point and the MDD, as well as between one MDD and another MDD. Those two protocols are SRTP [RFC3711] and DTLS

[RFC6347]. DTLS is included as our analysis [I-D.westerlund-perc-rtp-field-considerations] puts SRTP's design of leaving the RTP header fields in the clear into question in regards to use media confidentiality. To make third party attacks more difficult we would recommend using DTLS over SRTP for the hop-by-hop security.

5. Inband Key-Distribution

An EKT like inband key-distribution mechanism is assumed. This section discusses information that appear necessary to include in this security layer.

The unique source ID MUST be provied in EKT to prevent both denial of service attacks (Section 7.2.1) as well as splicing attacks (Section 7.2.4). The source ID also scopes the key sequence number. The key sequence number is monontionically increased each time the key distributed is changed. This is to prevent replay attacks including the EKT, that would update and replace the current key with an old key.

EKT could be used to provide other original field values that are assumed to have static mappings in MDD. Thus, original PT, Header Extension IDs could be provided.

While the current EKT mechanism is included in the RTP body of every packet, with a full EKT field sent periodically (e.g. every 100 ms), this may not be the optimal solution for PERC. The MDD will likely have a much better understanding of when an endpoint needs the full EKT field and may store and forward EKT when needed (new key sequence number or new receiving endpoint). This would not only save some bandwidth, but also minimize the time endpoints cannot decrypt because they have not got the latest key. Further optimizations could be to let the endpoint ack the reception of full EKT fields. Letting the control the delivery of full EKT fields can be done with the current EKT model where the full EKT fields are not bound to a specific SRTP packet, but only to a specific stream.

When the 32 bits Stream Specific End-to-End Sequence Nr is about to wrap, the sending end-point will have to rekey its transport key by sending a new full EKT field with a new transport key and a bumped up key sequence number. With easy rekeying, we note that 32-bits are sufficient also for really high bit-rates. At 3 Gbps using 1200 bytes of payload one need to rekey approximately every 3 hours.

6. Group Key-Management

In many cases the party controlling the PERC conference will want to limit the ability of participants to decrypt (and modify or inject) media produced before the participant joined or after the participant left. While some conferences will offer recording and allow all participants to decrypt the whole conference, participants modifying or injecting media after they have officially left the conference is not acceptable and must be mitigated. The known solution to this is to change the group EKT key. Either periodically or in conjunction with participants joining or leaving. After each change of the group EKT key, each sending endpoint needs to rekey also the transport key and deliver that to all remaining participants encrypted by the new group key.

7. Security Considerations

This section discusses various security considerations, especially a number of attacks.

7.1. Third Party Attacks

While an on-path third party attacker is always able to perform Denial of Service (DoS) Attacks by blocking all or selected packages, the PERC solution should be take measures to mitigate more serious DoS attacks form on-path and off-path attackers. 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 perform DoS attacks by connecting to different PERC entities and deliver Specificly crafted packets. One potential attack is if an attacker is able to get packets forwarded by the MDD, replacing a legitimate stream from one of the trusted endpoints. If hbh authentication Is not used, such an attack would only be detected in the receiving endpoints where the forged packets would be dropped. It is therefore essential that the MDD (or the call processing node) authenticates the endpoints as being invited members of the conference.

Another potential attack is a third party claiming to be an MDD, fooling endpoints points to send packets to the false MDD instead of the correct one. The deceived sending endpoint would then think the packets have been delivered to endpoints when they in fact have not been. If the false MDD can trick several endpoints to connect (or connect as an cascading MDD to another legitimate MDD) it may create a false version of the real conference, giving the connected endpoints a completely distorted view of the conference. To prevent

this attack all endpoints and MDDs MUST authenticate other MDDs to ensure that They are legitimate semi-trusted MDDs.

7.2. MDD Attacks

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

7.2.1. Denial of service

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

The MDD can also attempt perform resource consumption attacks on the receiving endpoint. One such attack would be to provide random SSRC/CSRC value to any RTP packet with an inband key-distribution message (EKT) attached. As the EKT message enables the receiver to form a new crypto context, the MDD can attempt to consume the receiving endpoints resources. This attack will be possible to detect and mitigate if the EKT messages contains the unique e2e stream id.

An denial of service attack is that the MDD rewrites the PT field to another codec. The MDD will usually know which PT corresponds to which codec. The effect of this attack is that an 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 degradations. For audio formats, especially sample based, this attack is likely to cause highly disturbing audio, that can be damaging to hearing and the playout equipment. This draft proposes that the original PT is provided end-to-end. However, without knowledge about the stream source's original media format MIME paraemters for each PT one can't verify correct mapping. Only detect attempts of remapping during the session.

7.2.2. Replay Attack

Replay attack is when an already received packets from a previous point in the RTP Stream is replayed as new packets. This could for example allow an MDD 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 jitter and network re-ordering window to be rejected. The end-to-end replay protection must be provided for the whole duration of the conference and must therefore based on a single monotonically

increasing number. The proposal in this document combines an end-to-end sequence number that is incremented with a key-sequence number, thus preventing that the reseting of the end-to-end sequence number when a re-keying occurs to allow old packets from being replayed.

7.2.3. Delayed Playout Attack

The delayed playout attack is an variant of the replay attack. This attack is possible even if e2e replay protection is in place. However, due to that the MDD is allowed to select a sub-set of streams and not forward the rest to a receiver the receiver has to accept gaps in the e2e packet sequence. The issue with this is that an MDD can select to not deliver a particular stream for a while. Within the window from last packet forward to the receiver and the latest received by the MDD, the MDD 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 by the transport delay.

To prevent this attack, we force the MDD to provide the receiver with the original RTP timestamp authenticated. Thus a receiver can compare the previously received sample's original timesamp with the original timestamp of the recently received sample. The timestamp difference should correspond to the difference in reception times with a maximum allowed variation corresponding to network jitter and a short fudge factor to enable the MDD to align different sources when acting as media switching mixer. Note this calculation will not function if the used RTP payload format switches and the different formats has different RTP timestamp rates. Thus the rules in [RFC7160] MUST be followed.

7.2.4. Splicing Attack

The splicing attack is an attack where a MDD receiving multiple media sources splices one media stream into the other. If the MDD would be able to change the SSRC without the receiver having any method for verifying the orignal source ID, then the MDD could first deliver stream A. And if the sequence numbers and other information allows it, the MDD can forward stream B under the same SSRC as stream A was previously forwarded.

This attack is mitigated by requiring each rtp stream to have unique source IDs that are provided to the receiver. That way the receiver would detect when the source ID switches for these streams.

7.2.5. Wrong Meta Data Attack

In case of several cascading MDDs, a malicious MDD may send forged meta data to another MDD, either giving the endpoints connected to the second MDD a modified view of what is happening in the conference (like who is speaking) or just degrading the quality of experience for endpoints connected to the second MDD. The false meta data could be any other hbh-protected fields. Especially in cases where two different conference providers or two different vendors of MDDs is involved in the conference, subtle forgeries meant to lower the experience for users of the competing service/MDD might be done.

Similar effect could result from honest MDDs having different algorithms, e.g. for selecting active speaker. Minor differences must likely be accepted as long as endpoints connected to different MDDs do not get very different view of what happened in the conference.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. References

9.1. Normative References

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

[I-D.ietf-avtcore-srtp-aes-gcm]
McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in Secure RTP (SRTP)", draft-ietf-avtcore-srtp-aes-gcm-17 (work in progress), June 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7160] Petit-Huguenin, M. and G. Zorn, Ed., "Support for Multiple Clock Rates in an RTP Session", RFC 7160, DOI 10.17487/RFC7160, April 2014, <http://www.rfc-editor.org/info/rfc7160>.

9.2. Informative References

[I-D.westerlund-perc-rtp-field-considerations] Westerlund, M., "Handling Considerations for the RTP fields in PERC", draft-westerlund-perc-rtp-field-considerations-00 (work in progress), October 2015.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

Authors' Addresses

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

Mats Naslund
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 33 739
Email: mats.naslund@ericsson.com

Magnus Westerlund
Ericsson
Farogatan 2
SE-164 80 Stockholm
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

M. Westerlund
J. Mattsson
Ericsson

July 6, 2015

WebRTC Use Case and Framework for Privacy Enhanced RTP Conferencing (PERC)
draft-westerlund-perc-webrtc-use-case-00

Abstract

The work so far on Privacy Enhanced RTP Conferencing, which allows end-to-end security also in centralized switch RTP based conference, has not considered WebRTC in detail. This document looks at the use case of WebRTC based endpoints, it also considers implications of using external providers for both conference applications and centralized media distribution devices. From this a number of challenges have been determined, and requirements are derived from these. Finally the draft presents some straw man for possible solutions.

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 http://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 January 7, 2016.

Table of Contents

1. Introduction

This document discusses the implications on PERC WG's work on providing end-to-end secure centralized RTP conferencing using WebRTC browsers as endpoints. The WebRTC environment contains a number of challenges that needs to be considered; these may affect how the final solution is designed. The authors have also have a strong interest in enabling usages where significant amount of sourcing of external resources are possible to perform. Not only the media distribution devices (MDD) and STUN/TURN resources, but also the core functionalities of the conference application, such as the find and connect to establish the conference. However, the control over the end-to-end security needs to be possible to maintain within a single organization. This organization needs to maintain control over both who is authorized to participate in a particular conference, as well as having control over the end-to-end keys used in that conference.

It needs to be stressed that the use case presented here is far from the only one where WebRTC endpoints could be used to establish a multiparty end-to-end secured conference. The authors have chose to focus on use case that combines WebRTC endpoints, contextual communication and outsourcing, a use case suitable for a number of enterprises, businesses, and services.

Section 2 goes through a possible use case and its high level motivation. Section 3 discusses the different trust domains as well as the entities that are considered in this use case. Section 4 discusses a number of challenges where several are unique to WebRTC compared to more native implementations of endpoints. Section 5 derives a number of requirements. Finally in Section 6 we present a straw man solution to some of the challenges we raised.

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 RFC 2119 [RFC2119].

2. The Use Case

This section discusses the representative use case in more detail as well as discussing relevant background information for this use case.

A use case (Enterprise Real-time contextual communication) is that an enterprise has the need for multi-party real-time conferencing with audio and video that is combined with enterprise's internal data that can be viewed and manipulated using a web application. The conferencing is intended to allow multiple employees or external consultants to discuss the data and manipulate its content as well as

present during the discussion, i.e. a form of contextual communication.

The already existing web application to view and manipulate the internal data is desirable to be able to re-use and the conference participants are already using a web browser for this purpose. Thus, basing the solution on WebRTC appears logical as that will enable integration of the Real-time communication (RTC) conferencing part with the existing web application.

The enterprise has no desire to maintain substantial RTC infrastructure to ensure well working conferencing, and prefers to source the needed services and components from external providers. However, the enterprise has strong interests in maintaining control of the security properties and ensure that its security goals are meet, it may even have legal requirements on its communication.

The enterprise already has existing methods for authenticating their employees, consultants and other external parties that have some access rights to the enterprise's data. It would be highly desirable to be able to re-use the existing user database and authorization verification.

3. Entities and Trust

This section discusses various entities and the trust in these roles.

3.1. Trust Domains

The entities belong to three different trust domains:

Trusted: Entities in the trusted domain are fully trusted to perform the role and actions put on them. They may have access to unprotected content and keying material used to protect content end-to-end.

Semi-trusted: Semi-trusted entities have no access to confidential material such as the content and the keying material used to protect content end-to-end. They are however trusted to perform basic operations for selective forwarding of content as well as session establishment.

Untrusted: Entities and functions in this domain are not trusted by the other entities participating in the conference or system. These entities are capable of preventing the conference from functioning, however.

3.2. Trusted Entities

The trusted entities that we consider in the use case are:

The Service Provider: The service provider is the organization that establishes the requirement for the end-to-end security in its RTC conference. It also has the control over entities that provide functionalities in the trusted domain for enabling the conference, such as the Key-Management Function and the Core Application. This represents the enterprise in the above use case description.

Conference Initiator: A human that schedules a conference. The conference contains a number of invited conference participants. These and only these participants shall be able to join this conference.

Conference Participant: A conference participant is a human or possibly another type of entity, like a conference room that has one or more identities that has been invited into the conference.

User Agent: The user agent is the WebRTC enabled endpoint that the conference participant uses to join the conference. In this document this is assumed to be a browser.

Core Application: The web application residing both in the client's server as well as the web based application in the User Agent. Should be able to delegate almost everything to the Conference Application. Non RTC content part of a contextual application is generally handled by this application.

Key Management Function: The function within the service provider that generates, stores and distributes the keys giving access to the conference to User Agents that can provide identity assertions that it can be verified and matches the invited participants.

Authorization Module: A sub-function of the Key Management Function (KMF) that verifies identity assertions according to a particular authorization method.

Conference Session Database: A sub-system of the KMF that contains information about scheduled conferences and which identities that have been invited by the Conference Initiator.

3.3. Semi-trusted Entities

The semi-trusted entities that we consider in the use case are:

Conference Provider: Provides the conferencing service, potentially from within a separate administrative domain than the service provider. This service contains find and connect functions to establish the conference, media distribution devices to selectively forward protected media content to the conference participants as well as support functions for media path establishment such as STUN and TURN services.

Conference Application: The web application residing both in the conference provider's servers as well in the User Agent that performs find and connect and other functions to establish and maintain the conference.

Media Distribution Devices: A device that performs switching and forwarding of protected media to the various conference participants' user agents or to another cascading media distribution device. The switching is performed based on meta information about the content that has been provided by the user agent. It applies security policies that prevent forwarding of traffic not originating with a user agent that is an invited participant. This protection is through a hop-by-hop security mechanism that provides integrity and source authentication as well as confidentiality of the metadata.

STUN/TURN: Support functions that help establish the transport path used to send media between a user agent and the media distribution device.

4. Challenges

This section discusses a number of challenges in meeting the goals as discussed in [I-D.jones-perc-private-media-reqts].

4.1. Enable Delegation

As described above, the Service Provider delegates the communication service to the Conference Provider. The Conference Provider may in turn delegate functions like the media distribution device and STUN/TURN services by sourcing them from other providers. Further, the infrastructure (servers and network) that these functions are run on, can also be sourced from external providers. This puts even higher demands on control and the ability to verify other entities actions from the perspective of the Service Provider.

The main security goal of providing end-to-end confidentiality across a centralized conferencing infrastructure is the main enabler of delegation, as the required trust in large part of the infrastructure

are significantly reduced by freeing them from handle any content as plain-text. However, that is not sufficient as not only the content handling needs to be limited to only the entities that are required to handle it. Also the key-management and authorization parts of the solution need to consider how they can limit the trust. For example the find and connect service is a semi-trusted part as it needs to be capable of establishing the connectivity with the right entities. However, the key-management and authorization system is the one that verifies the participants and their right to participate in a particular conference, and first then provides that participant's endpoint with the necessary secrets.

The system design needs to minimize the privacy sensitive information a particular functions needs. Thus, enabling as much functionality as possible to be outside of the trusted domain. Important functions in the semi-trusted domain, when so necessary to ensure secure operation of the system, must be verified by trusted entities.

4.2. Dealing with JavaScript related Attacks

The application, such as JavaScript application, running in the browser is a potential attack vector. Using various attacks, including cross-site scripting, the application can be compromised and perform the actions an attacker dictates. Even if the application running in the browser is malicious it must not be able to compromise the security of the conference, only perform denial of service attacks such as preventing the user from joining the conference.

A compromised application must be prevented from getting access to content. This will most likely mean that when using the end-to-end confidentiality, corresponding measures to prevent forwarding (of plain text content), access to raw data through APIs etc. that the media confidentiality mode defined in [I-D.ietf-rtcweb-alpn] have to be applied.

The compromised application will get access to who the peer(s) are in the conference. This is unpreventable as the application is the responsible for establishing the communication legs that is creating the conference. An attacker will also be able to use a compromised entity to forward protected content to a destination of its choice.

4.3. Forcing End-to-End Security

The Service Provider needs a method to ensure that when the conference provider application launches an RTCPeerConnections, they are forced to use end-to-end security with the keys provided by the KMF the Service Provider designates, and not the normal hop-by-hop

security only. Thus, the service provider needs a way of applying policies on an web application context. Policies that are inherited by any child contexts and which can't be modified by the application in the user agent.

4.4. Restricting Usage of the e2e Keying Material

While the RTCPeerConnection must use end-to-end security with the key provided by the Service Provider, neither the core or conference applications must be able to extract the key or even use the e2e key material for anything other than encrypted key transport (EKT). The user-agent will be required to have a secure key-store for the duration the key-material is present at the user-agent and valid. When the validity of the key-material expires the key-material needs to be disposed of to reduce the risk of retrospective attacks.

4.5. Enabling Flexible Authentication

The authorization methods should be flexible and enable different types of authorization back-ends. It is desirable that the method for authorization does not need to be implemented as part of the user agent. Requiring user agent modifications makes deployment of new authorization method cumbersome and difficult and open up for down grade attacks due to need for backwards compatibility support.

4.6. Binding Authorization to Endpoint

As the authorization will be used to retrieve the group key used to secure the RTP session end-to-end, it is important that the authorization is bound to the device and user agent where the user gave the authorization. Otherwise the conference provider would be able to move the authorization credentials to another endpoint, use that endpoint to retrieve the key and export it from that endpoint.

4.7. Securing Group Membership Changes

During an ongoing conference the set of participants participating in a conference will vary. In some usages it is important that joining participant can not use the received keying material to perform a retrospective attack and decrypt the content of the conference from a point prior to the participant joining. Nor should the participant after having left the conference be able continue to decrypt the content.

The known solution to this issue is to switch keys, both group key as well as the transport keys each endpoint uses to protect its streams. This puts certain requirements on key-management system. First the key-management system must track the current set of participants and

on changes initiate the change of the group key. This results in a second requirement that they key distribution method for the group key can handle asynchronous distribution events in the KMF to endpoint direction. Thirdly the transport key switching and distribution needs to handle non-synchronized switching by the different endpoints to the new group key.

A clear issue is how the KMF can ensure that a participant that is leaving is correctly accounted for and the key change happens in a timely fashion after the user left. First of all users may leave the conference abruptly due to severed communication or an endpoint that crashes. Secondly, the conference management application is only semi-trusted. The design will have to make choices on how to balance protocol complexities, resource consumption and achieved security properties.

An additional complexity with this mode of operation is that the conference participant likes to in a secure way know which other participants that currently are part of the conference. This information needs to be timely updated, and the current rooster needs to be authenticated to prevent attacks where participants are fooled to believe a particular participant has left, but is in fact still in the conference.

4.8. Key Revocation

The conference e2e group key is only required to reside on an endpoint for the duration its in use. That use is limited by the duration of the conference. When the conference ends there are no reasons to retain the key on the endpoint. Thus, when the conference ends it is desirable to have the key be revoked and deleted from the endpoint. This should be possible to initiate from the KMF when it learns that the conference has ended.

The user agent should upon the user closing the browsing context where the application runs deleting the keys to prevent their leakage.

5. Requirements

This section lists a number of derived requirements from the above challenges. The requirements are:

a. The Web application running in the User Agent MUST NOT be able to compromise the content confidentiality.

1. Including getting access to media content (raw or unencrypted) in the user agent through API or shared resources.

b. The conference provider's application (server as well as in the user agent) MUST NOT be able to downgrade the intended security properties and policies established by the service provider and the core application.

c. The key material for the end-to-end protection MUST NOT be possible to extract from any web application.

1. The user agent MUST protect the key-material against extraction by user or other software running on the same device.

2. The key material MUST be bound to the usage its intended to prevent leakage.

3. Upon the termination of the conference or the browsing context containing the application the key material SHALL be deleted

d. Different Authorization methods MUST be supported.

1. It's preferable that authorization methods can be supported without user agent modifications.

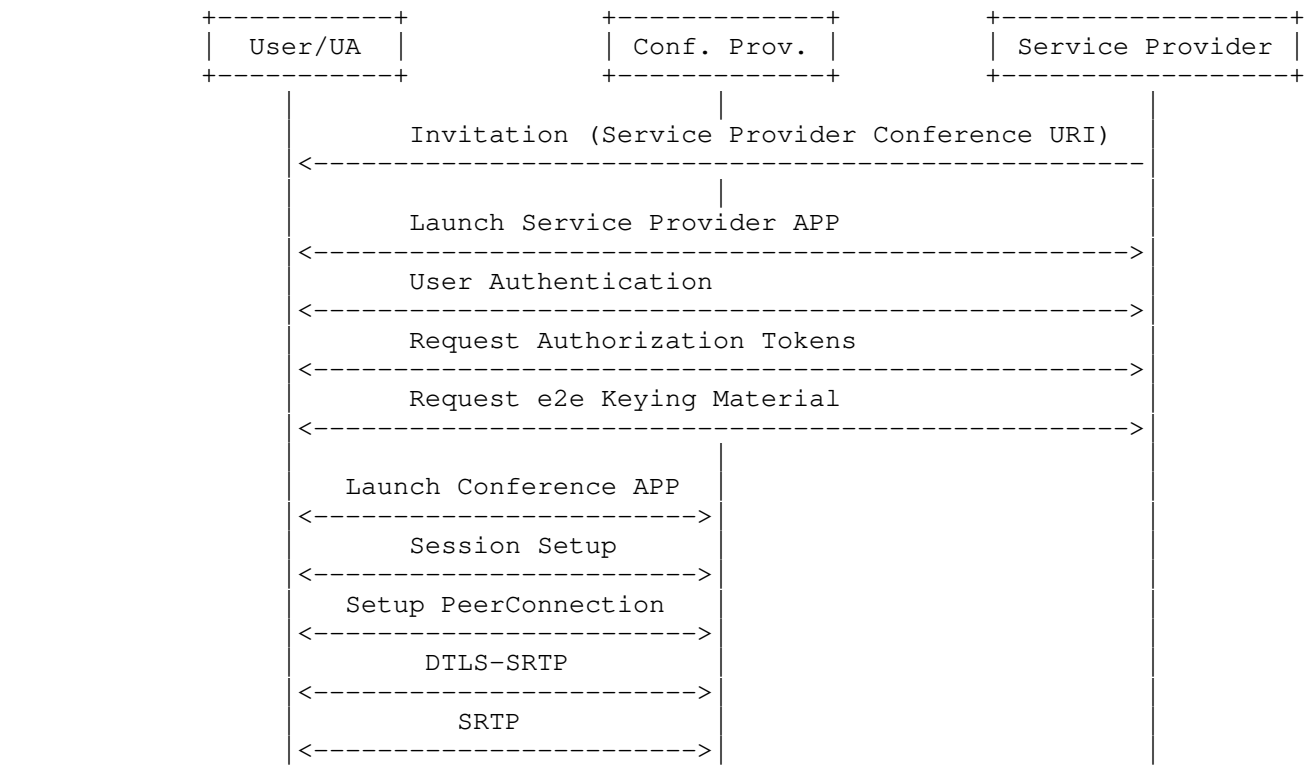
2. The authorization credentials MUST be bound to endpoint where the participant provided its credentials.

e. The design SHOULD support confidentiality where only the current set of participants has access to the media content.

6. Solution Straw-man

6.1. High Level Example Message Flows

The following figure shows a very high level illustration of an example message flow for Privacy Enhanced RTP Conferencing using WebRTC.



The Conference Initiator schedules a conference and invitations are sent out to the conference participants. This could for example be done via e-mail.

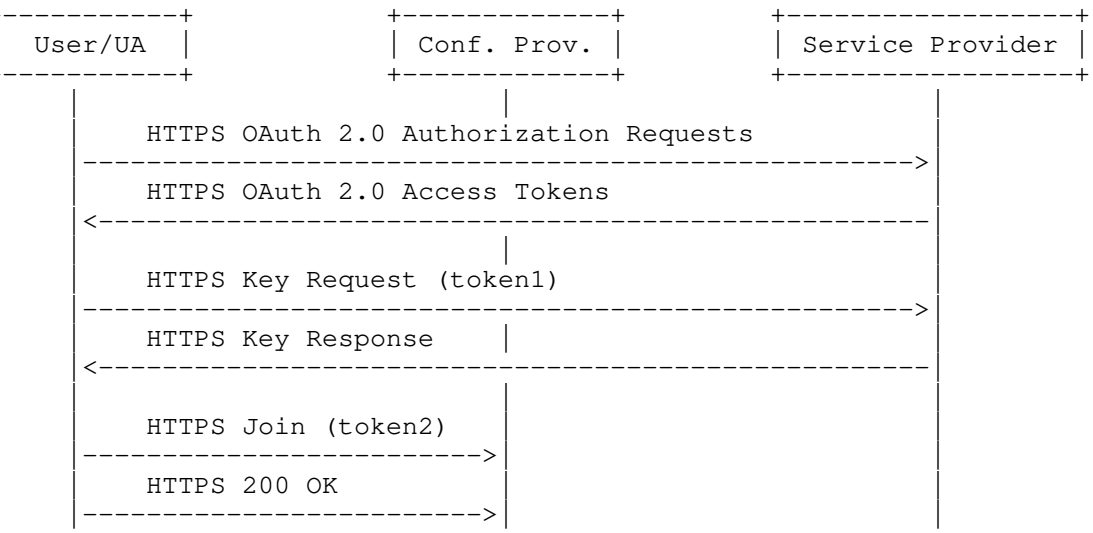
At a later stage, when the conference is about to start, the Conference Participant enters the conference URI in a browser it trusts to launch the core web application. The user then authenticates to the Service Provider Authorization Module (using a authentication method of choice), and downloads the end-to-end keying material from the Service Provider KMF.

The UA then launches the Conference Application, negotiates the session parameters and sets up the PeerConnection. The Conference provider validates that the participant is authorized by the Service Provider to join the conference. The hob-by-hop security is provided by DTLS-SRTP and SRTP (modified to handle end-to-end and hop-by-hop). The UA enforces the use of end-to-end security with the key provided by the Service Provider.

All communication except the invitation and the PeerConnection is to

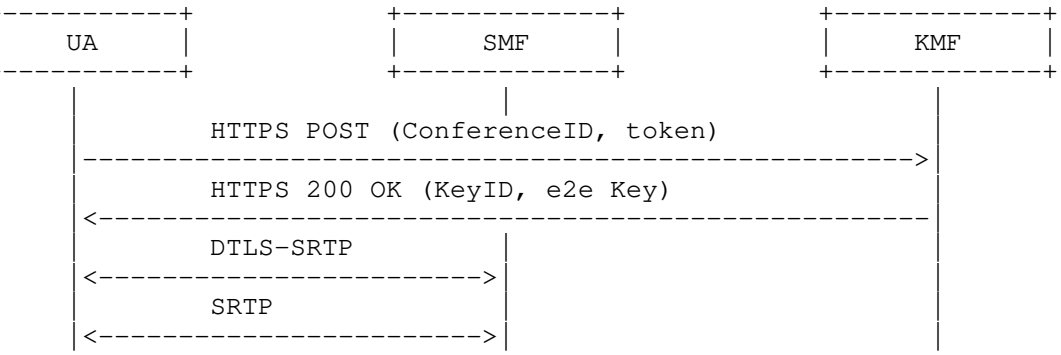
be done over HTTPS.

6.2. Authentication and Authorization



One way to make the end-to-end security solution flexible and enable integration with different types of Service Provider authorization back-ends is to use a general authorization framework such as OAuth 2.0. The User requests access tokens for all the protected resources from the Service Provider Authorization Module. The protected resources can be hosted by the Service Provider (e.g. the Key Management Function) as well as by the Conference provider. The use of OAuth 2.0 allows the same framework to be used in both cases.

6.3. Key Management



The core web application requests the end-to-end keying material from the Service Provider KMF. The successful HTTP response uses the HTTP Encryption-Key header [I-D.thomson-http-encryption] to distribute the end-to-end keying material to the UA. The new parameter "usage" and its value "EKT" instructs the UA that the keying material will be used with SRTP Encrypted Key Transport (EKT). The UA stores the keyid and the keying material for usage as the EKT Key. Key material received with the "usage=EKT" parameter SHALL NOT be extractable and SHALL only be used for EKT. The EKT processing MUST be handled by the UA.

An example successful HTTP response from the KMF is shown below:

```
HTTP/1.1 200 OK
Encryption-Key: keyid="pegh"; key="lupDujHomwIjlutebgharghmey";
usage="EKT"
Content-Length: 0
```

The hob-by-hop keying material is negotiated between the UA and the SMF using DTLS-SRTP.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

The whole document is about making WebRTC based cloud-based conferencing viable and trustworthy from a pervasive monitoring perspective.

9. Acknowledgements

The authors would like to thank Goeran AP Eriksson for challanging discussions and Russ White for valuable comments.

10. Informative References

[I-D.ietf-rtcweb-alpn]
Thomson, M., "Application Layer Protocol Negotiation for Web Real-Time Communications (WebRTC)",

draft-ietf-rtcweb-alpn-01 (work in progress), February 2015.

[I-D.jones-perc-private-media-reqts] Jones, P., Ismail, N., Benham, D., Buckles, N., Mattsson, J., Cheng, Y., and R. Barnes, "Requirements for Private Media in a Switched Conferencing Environment", draft-jones-perc-private-media-reqts-00 (work in progress), July 2015.

[I-D.thomson-http-encryption] Thomson, M., "Encrypted Content-Encoding for HTTP", draft-thomson-http-encryption-01 (work in progress), July 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

John Mattsson
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

M. Westerlund
J. Mattsson
Ericsson

October 19, 2015

WebRTC Use Case and Framework for Privacy Enhanced RTP Conferencing (PERC)
draft-westerlund-perc-webrtc-use-case-01

Abstract

The work so far on Privacy Enhanced RTP Conferencing, which allows end-to-end security also in centralized switched RTP based conferences, has not considered WebRTC in detail. This document looks at the use case of WebRTC based endpoints, it also considers implications of using external providers for both conference applications and centralized media distribution devices. From this a number of challenges have been determined, and requirements are derived from these. Finally the draft presents some straw man for possible solutions.

Table of Contents

1. Introduction

This document discusses the implications on PERC WG's work on providing end-to-end secure centralized RTP conferencing using WebRTC browsers as endpoints. The WebRTC environment contains a number of challenges that needs to be considered; these may affect how the final solution is designed. The authors have also have a strong interest in enabling usages where significant amount of sourcing of external resources are possible to perform. Not only the media distribution devices (MDD) and STUN/TURN resources, but also the core functionalities of the conference application, such as the find and connect to establish the conference. However, the control over the

end-to-end security needs to be possible to maintain within a single organization. This organization needs to maintain control over both who is authorized to participate in a particular conference, as well as having control over the end-to-end keys used in that conference.

It needs to be stressed that the use case presented here is far from the only one where WebRTC endpoints could be used to establish a multiparty end-to-end secured conference. The authors have chose to focus on use case that combines WebRTC endpoints, contextual communication and outsourcing, a use case suitable for a number of enterprises, businesses, and services.

Section 2 goes through a possible use case and its high level motivation. Section 3 discusses the different trust domains as well as the entities that are considered in this use case. Section 4 discusses a number of challenges where several are unique to WebRTC compared to more native implementations of endpoints. Section 5 derives a number of requirements. Finally in Section 6 we present a straw man solution to some of the challenges we raised.

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 RFC 2119 [RFC2119].

2. The Use Case

This section discusses the representative use case in more detail as well as discussing relevant background information for this use case.

A use case (Enterprise Real-time contextual communication) is that an enterprise has the need for multi-party real-time conferencing with audio and video that is combined with enterprise's internal data that can be viewed and manipulated using a web application. The conferencing is intended to allow multiple employees or external consultants to discuss the data and manipulate its content as well as present during the discussion, i.e. a form of contextual communication.

The already existing web application to view and manipulate the internal data is desirable to be able to re-use and the conference participants are already using a web browser for this purpose. Thus, basing the solution on WebRTC appears logical as that will enable integration of the Real-time communication (RTC) conferencing part with the existing web application.

The enterprise has no desire to maintain substantial RTC infrastructure to ensure well working conferencing, and prefers to source the needed services and components from external providers.

However, the enterprise has strong interests in maintaining control of the security properties and ensure that its security goals are meet, it may even have legal requirements on its communication.

The enterprise already has existing methods for authenticating their employees, consultants and other external parties that have some access rights to the enterprise's data. It would be highly desirable to be able to re-use the existing user database and authorization verification.

3. Entities and Trust

This section discusses various entities and the trust in these roles.

3.1. Trust Domains

The entities belong to three different trust domains:

Trusted: Entities in the trusted domain are fully trusted to perform the role and actions put on them. They may have access to unprotected content and keying material used to protect content end-to-end.

Semi-trusted: Semi-trusted entities have no access to confidential material such as the content and the keying material used to protect content end-to-end. They are however trusted to perform basic operations for selective forwarding of content as well as session establishment.

Untrusted: Entities and functions in this domain are not trusted by the other entities participating in the conference or system. These entities are capable of preventing the conference from functioning, however.

The entities in the trusted and semi-trusted domains are show in Figure 1, and described in more detail in Sections 3.2 and 3.3. Note that part of the "endpoint" is trusted, while other parts are only semi-trusted. A PERC conference involves more than one conference participant and may involve several MDDs.

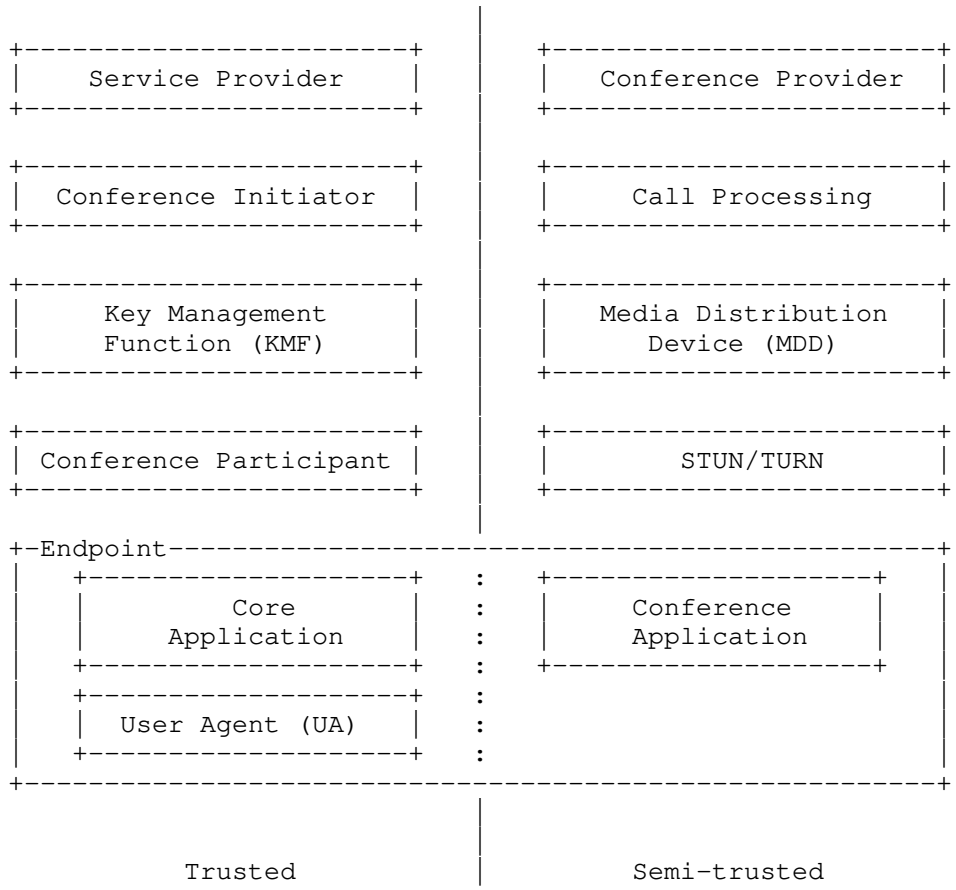


Figure 1: Entities in the Trusted and Semi-trusted Domains

3.2. Trusted Entities

The trusted entities that we consider in the use case are:

The Service Provider: The service provider is the organization that establishes the requirement for the end-to-end security in its RTC conference. It also has the control over entities that provide functionalities in the trusted domain for enabling the conference, such as the Key-Management Function and the Core Application. This represents the enterprise in the above use case description.

Conference Initiator: A human that schedules a conference. The conference contains a number of invited conference participants. These and only these participants shall be able to join this conference.

Conference Participant: A conference participant is a human or possibly another type of entity, like a conference room that has one or more identities that has been invited into the conference.

User Agent: The user agent is the WebRTC enabled endpoint that the conference participant uses to join the conference. In this document this is assumed to be a browser.

Core Application: The web application residing both in the service provider's server as well as the web based application in the User Agent. Should be able to delegate almost everything to the Conference Application. Non RTC content part of a contextual application is generally handled by this application.

Key Management Function: The function within the service provider that generates, stores and distributes the keys giving access to the conference to User Agents that can provide identity assertions that it can be verified and matches the invited participants.

Authorization Module: A sub-function of the Key Management Function (KMF) that verifies identity assertions according to a particular authorization method.

Conference Session Database: A sub-system of the KMF that contains information about scheduled conferences and which identities that have been invited by the Conference Initiator.

3.3. Semi-trusted Entities

The semi-trusted entities that we consider in the use case are:

Conference Provider: Provides the conferencing service, potentially from within a separate administrative domain than the service provider. This service contains find and connect functions to establish the conference, media distribution devices to selectively forward protected media content to the conference participants as well as support functions for media path establishment such as STUN and TURN services.

Conference Application: The web application residing both in the conference provider's servers as well in the User Agent that performs find and connect and other functions to establish and maintain the conference.

Media Distribution Devices: A device that performs switching and forwarding of protected media to the various conference participants' user agents or to another cascading media distribution device. The switching is performed based on meta

information about the content that has been provided by the user agent. It applies security policies that prevent forwarding of traffic not originating with a user agent that is an invited participant. This protection is through a hop-by-hop security mechanism that provides integrity and source authentication as well as confidentiality of the metadata.

STUN/TURN: Support functions that help establish the transport path used to send media between a user agent and the media distribution device.

4. Challenges

This section discusses a number of challenges in meeting the goals as discussed in [I-D.jones-perc-private-media-reqts].

4.1. Enable Delegation

As described above, the Service Provider delegates the communication service to the Conference Provider. The Conference Provider may in turn delegate functions like the media distribution device and STUN/TURN services by sourcing them from other providers. Further, the infrastructure (servers and network) that these functions are run on, can also be sourced from external providers. This puts even higher demands on control and the ability to verify other entities actions from the perspective of the Service Provider.

The main security goal of providing end-to-end confidentiality across a centralized conferencing infrastructure is the main enabler of delegation, as the required trust in large part of the infrastructure are significantly reduced by freeing them from handle any content as plain-text. However, that is not sufficient as not only the content handling needs to be limited to only the entities that are required to handle it. Also the key-management and authorization parts of the solution need to consider how they can limit the trust. For example the find and connect service is a semi-trusted part as it needs to be capable of establishing the connectivity with the right entities. However, the key-management and authorization system is the one that verifies the participants and their right to participate in a particular conference, and first then provides that participant's endpoint with the necessary secrets.

The system design needs to minimize the privacy sensitive information a particular functions needs. Thus, enabling as much functionality as possible to be outside of the trusted domain. Important functions in the semi-trusted domain, when so necessary to ensure secure operation of the system, must be verified by trusted entities.

4.2. Dealing with JavaScript related Attacks

The application, such as JavaScript application, running in the browser is a potential attack vector. Using various attacks, including cross-site scripting, the application can be compromised and perform the actions an attacker dictates. Even if the application running in the browser is malicious it must not be able to compromise the security of the conference, only perform denial of service attacks such as preventing the user from joining the conference.

A compromised application must be prevented from getting access to content. This will most likely mean that when using the end-to-end confidentiality, corresponding measures to prevent forwarding (of plain text content), access to raw data through APIs etc. that the media confidentiality mode defined in [I-D.ietf-rtcweb-alpn] have to be applied.

The compromised application will get access to who the peer(s) are in the conference. This is unpreventable as the application is the responsible for establishing the communication legs that is creating the conference. An attacker will also be able to use a compromised entity to forward protected content to a destination of its choice.

4.3. Forcing End-to-End Security

The Service Provider needs a method to ensure that when the conference provider application launches the RTCPeerConnections, they are forced to use end-to-end security with the keys provided by the KMF the Service Provider designates, and not normal hop-by-hop security only or end-to-end security with other keys. Thus, the service provider needs a way of applying policies on an web application context, or the conference participants must actively check and understand information in the browser chrome. This first approach could e.g. be done as the Service Provider web server setting policies and restrictions that the UA enforces towards the JavaScript. Policies that are inherited by any child contexts and which can’t be modified by the application in the user agent. A user clicking the correct link would then be secure. The second approach seems to give much weaker security as the average user do not look for security information and do not understand it. A desirable model is that of HTTPS, as long as an end user enters the correct URL, they are guaranteed e2e security.

4.4. Restricting Usage of the e2e Keying Material

While the RTCPeerConnection must use end-to-end security with the key provided by the Service Provider, neither the core or conference applications must be able to extract the key or even use the e2e key material for anything other than encrypted key transport (EKT) as this may lead to information leakage by e.g. so called two-time pad. The user-agent will be required to have a secure key-store for the duration the key-material is present at the user-agent and valid. When the validity of the key-material expires the key-material needs to be disposed of to reduce the risk of retrospective attacks.

4.5. Enabling Flexible Authentication

The authorization methods should be flexible and enable different types of authorization back-ends. It is desirable that the method for authorization does not need to be implemented as part of the user agent. Requiring user agent modifications makes deployment of new authorization method cumbersome and difficult and open up for down grade attacks due to need for backwards compatibility support.

4.6. Binding Authorization to Endpoint

As the authorization will be used to retrieve the group key used to secure the RTP session end-to-end, it is important that the authorization is bound to the device and user agent where the user gave the authorization. Otherwise the conference provider would be able to move the authorization credentials to another endpoint, use that endpoint to retrieve the key and export it from that endpoint.

4.7. Secure Binding of e2e Source Identifier to User Name or Alias

In many usages, it is important that the conference participants can see in real-time who is participating and who is talking. This requires that the endpoint can map the e2e source identifier to the user name. The list of participant names as well as the binding to the e2e source identifiers needs to be authenticated by a trusted party to prevent attacks where an semi-trusted entity suggests an incorrect binding between an e2e Source Identifier and a user name.

4.8. Securing Group Membership Changes

During an ongoing conference the set of participants participating in a conference will vary. In some usages a late joining participant should get access to keying material to decrypt a conference recording. In other usages it is important that joining participant can not use the received keying material to perform a retrospective attack and decrypt the content of the conference from a point prior

to the participant joining. Nor should the participant after having left the conference be able continue to decrypt the content.

The known solution to this issue is to switch keys, both group key as well as the transport keys each endpoint uses to protect its streams. This puts certain requirements on key-management system. First the key-management system must track the current set of participants and on changes initiate the change of the group key. This results in a second requirement that they key distribution method for the group key can handle asynchronous distribution events in the KMF to endpoint direction. Thirdly the transport key switching and distribution needs to handle non-synchronized switching by the different endpoints to the new group key.

A clear issue is how the KMF can ensure that a participant that is leaving is correctly accounted for and the key change happens in a timely fashion after the user left. First of all users may leave the conference abruptly due to severed communication or an endpoint that crashes. Secondly, the conference management application is only semi-trusted. The design will have to make choices on how to balance protocol complexities, resource consumption and achieved security properties.

An additional complexity with this mode of operation is that the conference participant likes to in a secure way know which other participants that currently are part of the conference. This information needs to be timely updated, and the current rooster needs to be authenticated to prevent attacks where participants are fooled to believe a particular participant has left, but is in fact still in the conference.

4.9. Key Revocation

The conference e2e group key is only required to reside on an endpoint for the duration its in use. That use is limited by the duration of the conference. When the conference ends there are no reasons to retain the key on the endpoint. Thus, when the conference ends it is desirable to have the key be revoked and deleted from the endpoint. This should be possible to initiate from the KMF when it learns that the conference has ended.

The user agent should upon the user closing the browsing context where the application runs deleting the keys to prevent their leakage.

5. Requirements

This section lists a number of derived requirements from the above challenges. The requirements are:

a. The Web application running in the User Agent MUST NOT be able to compromise the content confidentiality.

    1. Including getting access to media content (raw or unencrypted) in the user agent through API or shared resources.

b. The conference provider's application (server as well as in the user agent) MUST NOT be able to downgrade the intended security properties and policies established by the service provider and the core application.

c. The key material for the end-to-end protection MUST NOT be possible to extract from any web application.

    1. The user agent MUST protect the key-material against extraction by user or other software running on the same device.

    2. The key material MUST be bound to the usage its intended to prevent leakage.

    3. Upon the termination of the conference or the browsing context containing the application the key material SHALL be deleted

d. Different Authorization methods MUST be supported.

    1. It's preferable that authorization methods can be supported without user agent modifications.

    2. The authorization credentials MUST be bound to endpoint where the participant provided its credentials.

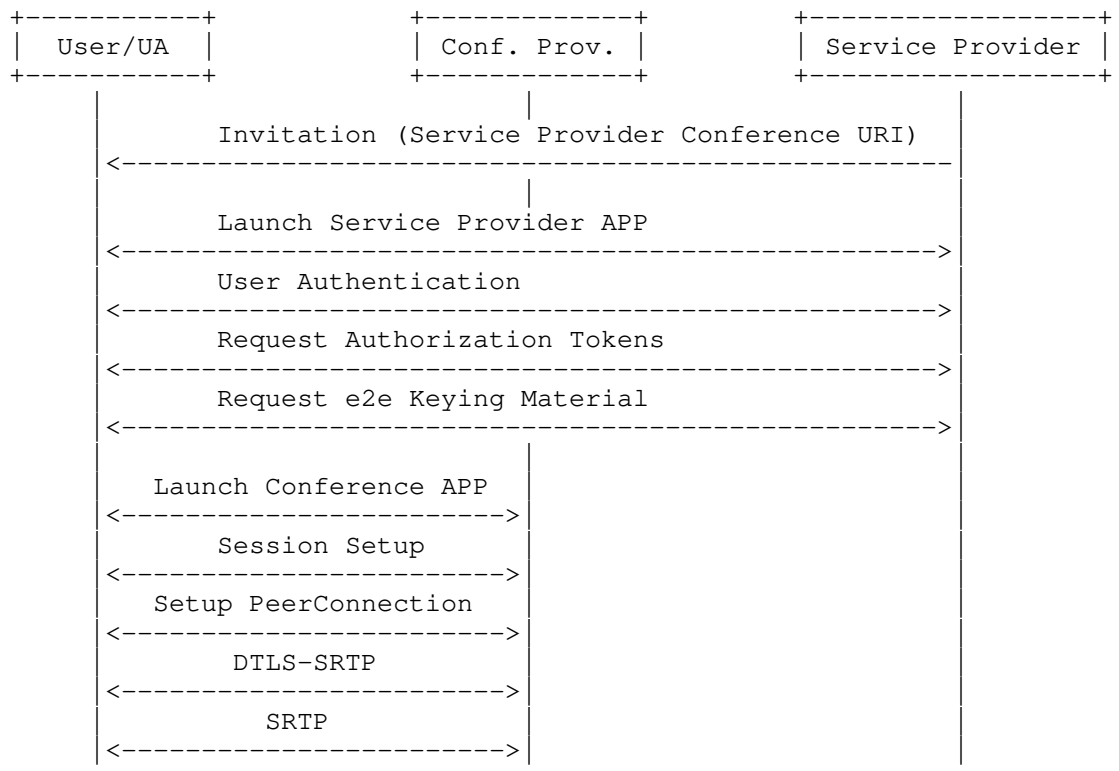
e. The design SHOULD support confidentiality where only the current set of participants has access to the media content.

f. The conference rooster and the binding to the e2e source identifies MUST be provided by a trusted party.

6. Solution Straw-man

6.1. High Level Example Message Flows

The following figure shows a very high level illustration of an example message flow for Privacy Enhanced RTP Conferencing using WebRTC.



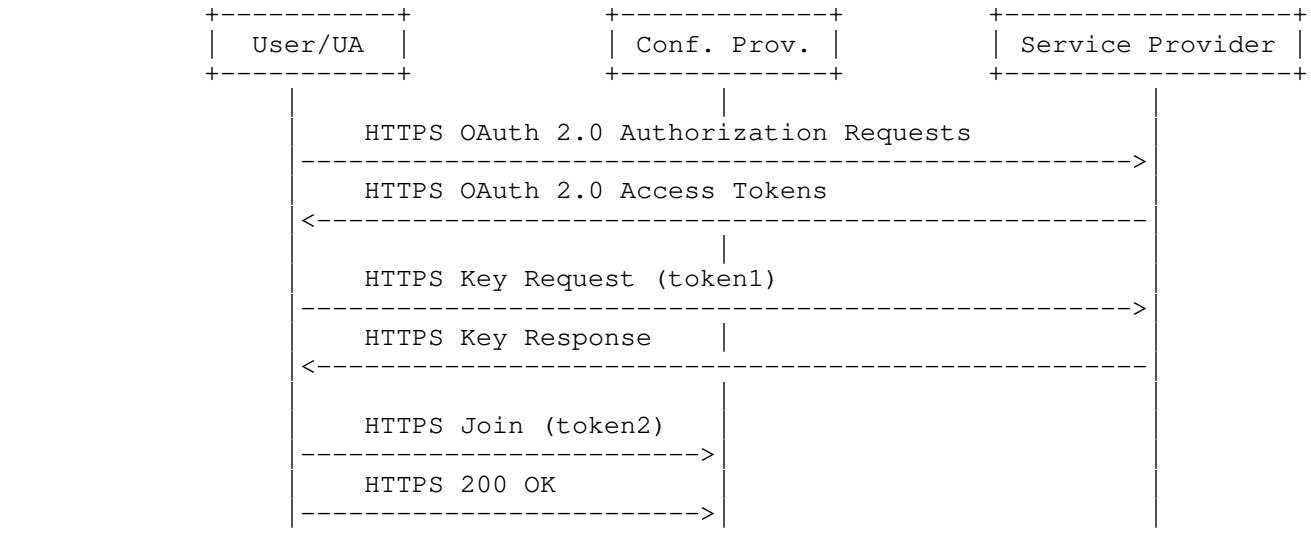
The Conference Initiator schedules a conference and invitations are sent out to the conference participants. This could for example be done via e-mail.

At a later stage, when the conference is about to start, the Conference Participant enters the conference URI in a browser it trusts to launch the core web application. The user then authenticates to the Service Provider Authorization Module (using a authentication method of choice), and downloads the end-to-end keying material from the Service Provider KMF.

The UA then launches the Conference Application, negotiates the session parameters and sets up the PeerConnection. The Conference provider validates that the participant is authorized by the Service Provider to join the conference. The hob-by-hop security is provided by DTLS-SRTP and SRTP (modified to handle end-to-end and hop-by-hop). The UA enforces the use of end-to-end security with the key provided by the Service Provider.

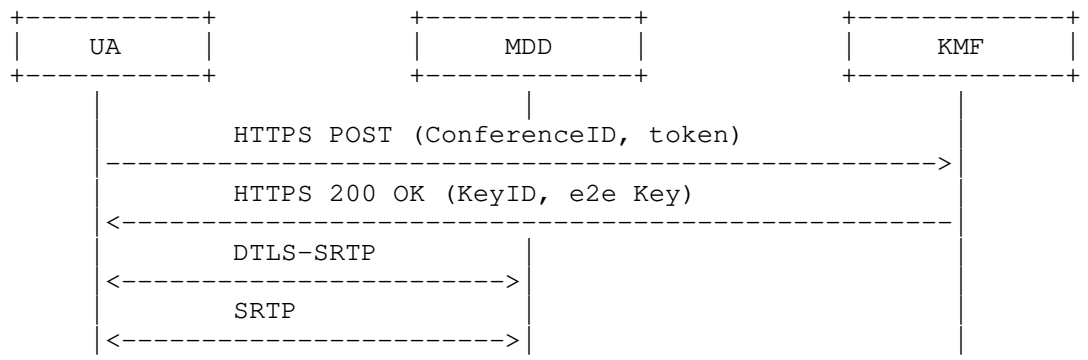
All communication except the invitation and the PeerConnection is to be done over HTTPS.

6.2. Authentication and Authorization



One way to make the end-to-end security solution flexible and enable integration with different types of Service Provider authorization back-ends is to use a general authorization framework such as OAuth 2.0. The User requests access tokens for all the protected resources from the Service Provider Authorization Module. The protected resources can be hosted by the Service Provider (e.g. the Key Management Function) as well as by the Conference provider. The use of OAuth 2.0 allows the same framework to be used in both cases.

6.3. Key Management



The core web application requests the end-to-end keying material from the Service Provider KMF. The successful HTTP response uses the HTTP Encryption-Key header [I-D.thomson-http-encryption] to distribute the end-to-end keying material to the UA. The new parameter "usage" and its value "EKT" instructs the UA that the keying material will be used with SRTP Encrypted Key Transport (EKT). The UA stores the keyid and the keying material for usage as the EKT Key. Key material received with the "usage=EKT" parameter SHALL NOT be extractable and SHALL only be used for EKT. The EKT processing MUST be handled by the UA.

An example successful HTTP response from the KMF is shown below:

```
HTTP/1.1 200 OK
Encryption-Key: keyid="pegh"; key="lupDujHomwIjlutebgharghmey";
usage="EKT"
Content-Length: 0
```

The hob-by-hop keying material is negotiated between the UA and the MDD using DTLS-SRTP.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

The whole document is about making WebRTC based cloud-based conferencing viable and trustworthy from a pervasive monitoring perspective.

9. Acknowledgements

The authors would like to thank Goeran AP Eriksson for challanging discussions and Russ White for valuable comments.

10. Informative References

[I-D.ietf-rtcweb-alpn]
Thomson, M., "Application Layer Protocol Negotiation for Web Real-Time Communications (WebRTC)", draft-ietf-rtcweb-alpn-01 (work in progress), February 2015.

[I-D.jones-perc-private-media-reqts]
Jones, P., Ismail, N., Benham, D., Buckles, N., Mattsson, J., and R. Barnes, "Private Media Requirements in Privacy Enhanced RTP Conferencing", draft-jones-perc-private-media-reqts-00 (work in progress), July 2015.

[I-D.thomson-http-encryption]
Thomson, M., "Encrypted Content-Encoding for HTTP", draft-thomson-http-encryption-02 (work in progress), October 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

John Mattsson
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com