AVT
Internet-Draft
Intended status: Informational
Expires: September 3, 2011

A. Begen
Cisco

March 2, 2011

Considerations and Guidelines for Deploying the Rapid Acquisition of Multicast RTP Sessions (RAMS) Method
draft-begen-avtext-rams-scenarios-00

Abstract

The Rapid Acquisition of Multicast RTP Sessions (RAMS) solution is a method based on RTP and RTP Control Protocol (RTCP) that enables an RTP receiver to rapidly acquire and start consuming the RTP multicast data. Upon a request from the RTP receiver, an auxiliary unicast RTP retransmission session is set up between a retransmission server and the RTP receiver, over which the reference information about the new multicast stream the RTP receiver is about to join is transmitted at an accelerated rate. This often precedes, but may also accompany, the multicast stream itself. When there is only one multicast stream to be acquired, the RAMS solution works in a straightforward manner. However, when there are two or more multicast streams to be acquired from the same or different multicast RTP sessions, care should be taken to configure each RAMS session appropriately. This document provides example scenarios and offers guidelines.

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 September 3, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Rapid Acquisition of Multicast RTP Sessions (RAMS) solution is a method based on RTP and RTP Control Protocol (RTCP) that enables an RTP receiver to rapidly acquire and start consuming the RTP multicast data. Through an auxiliary unicast RTP retransmission session [RFC4588], the RTP receiver receives a reference information about the new multicast stream it is about to join. This often precedes, but may also accompany, the multicast stream itself. The RAMS solution is documented in detail in [I-D.ietf-avt-rapid-acquisition-for-rtp].

The RAMS specification [I-D.ietf-avt-rapid-acquisition-for-rtp] has provisions for concurrently acquiring multiple streams inside a multicast RTP session. However, the specification has mostly focused on the simplest case, which is when the RTP receiver acquires only one multicast stream at a time, to explain the protocol details.

There are certain deployment models where a multicast RTP session may have two or more multicast streams associated with it. There are also cases, where an RTP receiver may be interested in acquiring one or more multicast streams from several multicast RTP sessions. In scenarios where multiple RAMS sessions will be simultaneously run by an RTP receiver, they need to be coordinated. In this document, we present scenarios from real-life deployments and provide guidelines.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Editor’s note: I am inclined not use any 2119 keyword in this document and remove this section altogether.

3. Background

In the following discussion, we assume that there are two RTP streams (1 and 2) that are somehow associated with each other. These could be audio and video elementary streams for the same TV channel, or they could be an MPEG2-TS stream (that has audio and video multiplexed together) and its Forward Error Correction (FEC) stream.

It is important to note that a source-specific multicast (SSM) session is defined by its (distribution) source address and

(destination) multicast group and there can be only one feedback target per SSM session [RFC5760]. So, if the RTP streams are distributed by different sources or over different multicast groups, they are considered different SSM sessions. While different SSM sessions can normally share the same feedback target address and/or port, RAMS requires each unique feedback target (i.e., the combination of address and port) to be associated with at most one RTP session (See Section 6.2 of [I-D.ietf-avt-rapid-acquisition-for-rtp]).

Two or more multicast RTP streams can be transmitted in the same RTP session (i.e., in a single UDP flow). This is called Synchronization Source (SSRC) multiplexing. In this case, (de)multiplexing is done at the SSRC level. Alternatively, the multicast RTP streams can be transmitted in different RTP sessions (i.e., in different UDP flows), which is called session multiplexing. In practice, there are different deployment models for each multiplexing scheme.

Generally, two different media streams with different clock rates are suggested to use different SSRCs or to be carried in different RTP sessions to avoid complications in RTCP reports. Some of the fields in RAMS messages might depend on the clock rate. Thus, in a single RTP session, RTP streams carrying payloads with different clock rates need to have different SSRCs. Since RTP streams in the same RTP session but with different SSRCs do not share the sequence numbering, each stream needs to be acquired individually.

In the remaining sections, only the relevant portions of the SDP descriptions [RFC4566] will be provided. For an example of a full SDP description, refer to Section 8.3 of [I-D.ietf-avt-rapid-acquisition-for-rtp].

4. Example Scenarios

4.1. Scenario #1: Two Multicast Groups

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over different multicast groups. A practical use case is where the first and second SSM sessions carry the primary video stream and its associated FEC stream, respectively.

We run an individual RAMS session for each of these RTP streams that we want to rapidly acquire. These RAMS sessions can be run in parallel. If they are, the RTP receiver needs to pay attention to using the shared bandwidth appropriately among the two unicast bursts. As explained earlier, there has to be a different feedback target for these two SSM sessions.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40000 RTP/AVPF 97
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtcp:42000 IN IP4 192.0.2.1
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

Note that the multicast destination ports in the above SDP do not matter, and they could be the same or different. The "FEC-FR" grouping semantics are defined in [RFC5956].

4.2. Scenario #2: One Multicast Group

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over the same multicast group with different destination ports. A practical use case is where the SSM session carries the primary video and audio streams, each destined to a different port.

Similar to scenario #1, we run individual RAMS sessions for each RTP stream that we want to rapidly acquire (Note that the RAMS request sent by an RTP receiver could indicate the desire to acquire all or a subset or one of the available RTP streams in an SSM session). Compared to the previous scenario, the only difference is that in this case the join times for both streams need to be coordinated as they are on the same multicast session.

```
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=audio 40001 RTP/AVPF 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:2 cname:ch1_audio@example.com
a=mid:Channel1_Audio
```

Note that the destination ports in the above SDP need to be distinct per [RFC5888].

If RTP streams 1 and 2 share the same distribution source, then there is only one SSM session, which means that there can be only one feedback target (as shown in the SDP description above). This requires RTP streams 1 and 2 to have their own unique SSRC values (also as shown in the SDP description above). If RTP streams 1 and 2 do not share the same distribution source, meaning that their respective SSM sessions can use different feedback target transport addresses, then their SSRC values do not have to be different from each other.

4.3. Scenario #3: SSRC Multiplexing

This is the scenario for SSRC multiplexing where both RTP streams are transmitted over the same multicast group to the same destination port. This is a less practical scenario but it could be used where the SSM session carries both the primary video and audio stream, destined to the same port.

Similar to scenario #2, we run individual RAMS sessions and the join time needs to be coordinated. In this case, there is only one distribution source and the destination multicast address is shared. Thus, there is always one SSM session and one feedback target.

```
m=video 40000 RTP/AVPF 96 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=ssrc:2 cname:ch1_audio@example.com
a=mid:Channel1
```

4.4. Scenario #4: Payload-Type Multiplexing

This is the scenario for payload-type multiplexing.

In this case, instead of two, we have only one RTP stream (and one RTP session) carrying both payload types (e.g., media payload and its FEC data). While this scheme is permissible per [RFC3550], it has several drawbacks. For example, RTP packets carrying different payload formats will share the same sequence numbering space, and the retransmission and RAMS operations will not be able to be applied based on the payload type. For other drawbacks and details, see Section 5.2 of [RFC3550].

5. Feedback Target and SSRC Signaling Issues

The RAMS protocol uses the common packet format from [RFC4585], which has a field to signal the media sender SSRC. The SSRCs for the RTP streams can be signaled out-of-band in the SDP, or could be learned from the RTP packets once the transmission starts. In RAMS, the latter cannot be used.

Signaling the media sender SSRC value helps the feedback target correctly identify the RTP stream to be acquired. If a feedback target is serving multiple SSM sessions on a particular port, all the RTP streams in these SSM sessions are supposed to have a unique SSRC value. However, since this is not an easy requirement to satisfy, RAMS specification forbids to have more than one RTP session to be associated with a specific feedback target.

6. FEC during RAMS and Bandwidth Issues

Suppose that RTP stream 1 denotes the primary video stream that has a bitrate of 10 Mbps and RTP stream 2 denotes the FEC stream that has a bitrate of 1 Mbps. Also assume that the RTP receiver knows that it can receive data at a maximum bitrate of 22 Mbps. SDP can specify the bitrate ("b=" line in Kbps) of each media session (per "m" line).

6.1. Scenario #1

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over different multicast groups.

This is the preferred deployment model for FEC. Having FEC in a different multicast group provides flexibility for not only the RTP receivers that are not FEC capable but also the ones that are FEC capable but are not willing to receive FEC during the rapid acquisition.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
b=TIAS:10000
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40000 RTP/AVPF 97
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtcp:42000 IN IP4 192.0.2.1
a=rtpmap:97 1d-interleaved-parityfec/90000
b=TIAS:1000
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

If the RTP receiver does not want to receive FEC until the acquisition of the primary video stream is completed, the total available bandwidth can be used for faster acquisition of the primary video stream. In this case, the RTP receiver can request a Max Receive Bitrate of 22 Mbps in the RAMS Request message. Once RAMS has been completed, the RTP receiver can join the FEC multicast session, if desired.

If the RTP receiver wants to rapidly acquire both primary and FEC streams, it needs to allocate the total bandwidth among the two RAMS sessions and indicate individual Max Receive Bitrate values in each respective RAMS Request message. Since less bandwidth will be used to acquire the primary video stream, the acquisition of the primary video session will take a longer time on the average.

While the RTP receiver can update the Max Receive Bitrate values during the course of the RAMS session, this approach is more error-prone due to the possibility of losing the update messages.

6.2. Scenario #2

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over the same multicast group with different destination ports.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
b=TIAS:10000
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40001 RTP/AVPF 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:97 1d-interleaved-parityfec/90000
b=TIAS:1000
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

Similar to scenario #1, the RTP receiver can first ask for RAMS for the primary video stream at the full receive bitrate. But, upon the multicast join, the available bandwidth for the burst drops to 11 Mbps instead of 12 Mbps. Regardless of whether FEC is desired or not by the RTP receiver, its bitrate needs to be taken into account once the RTP receiver joins the SSM session.

If the RTP receiver wants to rapidly acquire both primary and FEC streams, the same conditions explained for scenario #1 apply. The only difference from scenario #1 is that here the join time is the same for both the primary video and FEC streams.

6.3. Scenario #3

This is the scenario for SSRC multiplexing where both RTP streams are transmitted over the same multicast group to the same destination port.

```
m=video 40000 RTP/AVPF 96 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
a=rtpmap:97 1d-interleaved-parityfec/90000
a=fmtp:97 L=10; D=10; repair-window=200000
a=ssrc:1 cname:ch1_video@example.com
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1
b=TIAS:11000
a=mid:Channel1
```

This is similar to scenario #2. However, since we cannot explicitly specify the bitrates for the primary and FEC streams, the RTP receiver can request to rapidly acquire both streams in parallel. In this case, two separate RAMS Request messages have to be sent by the RTP receiver to the feedback target.

Note that based on the "a=fmtp" line for the FEC stream, it could be possible to infer the bitrate of this FEC stream and set the Max Receive Bitrate value accordingly.

7. Security Considerations

There are no security considerations in this document.

8. IANA Considerations

There are no IANA considerations in this document.

9. Acknowledgments

I would like to thank various individuals in the AVT and MMUSIC WGs, and my friends at Cisco for their comments and feedback.

10. References

10.1. Normative References

[I-D.ietf-avt-rapid-acquisition-for-rtp]
Steeg, B., Begen, A., Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", draft-ietf-avt-rapid-acquisition-for-rtp-17 (work in

progress), November 2010.

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

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

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

10.2. Informative References

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.

Author’s Address

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
Canada

Email: abegen@cisco.com

AVT
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2011

J. Lennox, Ed.
Vidyo
E. Ivov
Jitsi
E. Marocco
Telecom Italia

March 14, 2011

A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication
draft-ietf-avtext-client-to-mixer-audio-level-01

Abstract

This document defines a mechanism by which packets of Real-Time Transport Protocol (RTP) audio streams can indicate, in an RTP header extension, the audio level of the audio sample carried in the RTP packet. In large conferences, this can reduce the load on an audio mixer or other middlebox which wants to forward only a few of the loudest audio streams, without requiring it to decode and measure every stream that is received.

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 September 15, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

In a centralized Real-Time Transport Protocol (RTP) [RFC3550] audio conference, an audio mixer or forwarder receives audio streams from many or all of the conference participants. It then selectively forwards some of them to other participants in the conference. In large conferences, it is possible that such a server might be receiving a large number of streams, of which only a few should be forwarded to the other conference participants.

In such a scenario, in order to pick the audio streams to forward, a centralized server needs to decode, measure audio levels, and possibly perform voice activity detection on audio data from a large number of streams. The need for such processing limits the size or number of conferences such a server can support.

As an alternative, this document defines an RTP header extension [RFC5285] through which senders of audio packets can indicate the audio level of the packets' payload, reducing the processing load for a server.

The header extension in this draft is different to, but complementary with, the one defined in [I-D.ietf-avtext-mixer-to-client-audio-level], which defines a mechanism by which audio mixers can indicate to clients the levels of the contributing sources that made up the mixed audio.

2. 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] and indicate requirement levels for compliant implementations.

3. Audio Levels

The audio level header extension carries both the level of the audio carried in the RTP payload of the packet it is associated with, as well as an indication as to whether voice activity has been detected in the packet.

The form of the audio level extension block is as follows:

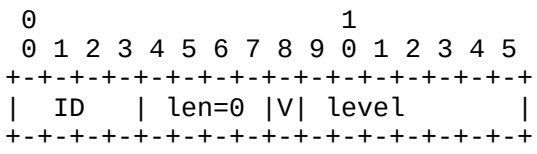

Figure 1

The length field takes the value 0 to indicate that 1 byte follows.

The audio level is defined in the same manner as is audio noise level in the RTP Comfort Noise [RFC3389] specification. In that specification, the overall magnitude of the noise level is encoded into the first byte of the payload, with spectral information about the noise in subsequent bytes. This specification's audio level parameter is defined so as to be identical to the comfort noise payload's noise-level byte.

The magnitude of the audio level is packed into the seven least significant bits of the single byte of the header extension, shown in Figure 1. The least significant bit of the audio level magnitude is packed into the least significant bit of the byte. The most significant bit of the byte is used as a separate flag bit "V", defined below.

The audio level is expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov. dBov is the level, in decibels, relative to the overload point of the system, i.e. the maximum-amplitude signal that can be handled by the system without clipping. (Note: Representation relative to the overload point of a system is particularly useful for digital implementations, since one does not need to know the relative calibration of the analog circuitry.) For example, in the case of u-law (audio/pcmu) audio [ITU.G711.1988], the 0 dBov reference would be a square wave with values +/- 8031. (This translates to 6.18 dBm0, relative to u-law's dBm0 definition in Table 6 of G.711.)

The reference implementation section in [I-D.ietf-avtext-mixer-to-client-audio-level] provides a sample implementation of an audio level calculator that helps obtain such values from raw audio samples.

In addition, a flag bit (labeled V) indicates whether the encoder believes the audio packet contains voice activity (1) or does not (0). The voice activity detection algorithm is unspecified and left implementation-specific.

The audio level for digital silence (e.g. all-0 pcmu audio), for example for a muted audio source, MAY be represented as 127 (-127 dBov), regardless of the dynamic range of the encoded audio format.

When this header extension is used with RTP data sent using the RTP Payload for Redundant Audio Data [RFC2198], the header's data describes the contents of the primary encoding.

4. Signaling (Setup) Information

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdrext:audio-level". There is no additional setup information needed for this extension (no extensionattributes).

5. Considerations on Use

Mixers and forwarders generally should not base audio forwarding decisions directly on packet-by-packet audio level information, but rather should apply some analysis of the audio levels and trends. This general rule applies whether audio levels are provided by endpoints (as defined in this document), or are calculated at a server, as would be done in the absence of this information. This section discusses several issues that mixers and forwarders may wish to take into account. (Note that this section provides design guidance only, and is not normative.)

First of all, audio levels should generally be measured over longer intervals than that of a single audio packet. In order to avoid false-positives for short bursts of sound (such as a cough or a dropped microphone), it is often useful to require that a participant's audio level be maintained for some period of time before considering it to be "real", i.e. some type of low-pass filter should be applied to the audio levels. Note, though, that such filtering must be balanced with the need to avoid clipping of the beginning of a speaker's speech.

Additionally, different participants may have their audio input set differently. It may be useful to apply some sort of automatic gain control to the audio levels. There are a number of possible approaches to acheiving this, e.g. by measuring peak audio levels, by average audio levels during speech, or by measuring background audio levels (average audio level levels during non-speech).

6. Limitations

The audio levels carried by the extension header defined by this document are defined as dBov, decibels below system overload.

In principle, it could be more useful to have, instead, dB SPL, decibels of sound pressure level. In traditional telephony systems, telephone handsets were calibrated such that a particular (e.g.) u-law audio level, or analog voltage, corresponded to a particular sound pressure level at the handset's mouthpiece.

However, in many environments, this information is not available. Notably, PC soundcard hardware can only determine the levels of mic- or line-in at the hardware input, and operating systems usually allow further adjustments of audio input levels without providing information about these transformations to applications. Furthermore, in many circumstances, such as speech synthesis or mixed audio, an "audio" signal may in fact never have actually existed as sound pressure at all.

Thus, while information about the correspondance between dB SPL and dBov, or encoded audio, could be useful, this document does not attempt to define it. If there are circumstances in which this information would be useful, a separate header extension would be straightforward to define. (The information carried by such a header extension could indeed be useful independently from the information in the header extension defined by this document.)

7. Security Considerations

A malicious endpoint could choose to set the values in this header extension falsely, so as to falsely claim that audio or voice is or is not present. It is not clear what could be gained by falsely claiming that audio is not present, but an endpoint falsely claiming that audio is present could perform a denial-of-service attack on an audio conference, so as to send silence to suppress other conference members' audio. Thus, a device relying on audio level data from untrusted endpoints SHOULD periodically audit the level information transmitted, taking appropriate corrective action if endpoints appear to be sending incorrect data. (Note that endpoints MAY choose to measure audio levels prior to encoding, so some degree of discrepancy SHOULD be tolerated.)

In the Secure Real-Time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but not encrypted. When this header extension is used, audio levels are therefore visible on a packet-by-packet basis to an attacker passively observing the audio

stream. As discussed in [I-D.perkins-avt-srtp-vbr-audio], such an attacker might be able to infer information about the conversation, possibly with phoneme-level resolution. In scenarios where this is a concern, additional mechanisms SHOULD be used to protect the confidentiality of the header extension. One solution would be header extension encryption [I-D.lennox-avt-srtp-encrypted-extension-headers].

8. IANA Considerations

This document defines a new extension URI to the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdrext:audio-level
Description: Audio Level
Contact: jonathan@vidyo.com
Reference: RFC XXXX

9. References

9.1. Normative References

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

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 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.

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

9.2. Informative References

[I-D.ietf-avtext-mixer-to-client-audio-level] Ivov, E., Marocco, E., and J. Lennox, "A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", draft-ietf-avtext-mixer-to-client-audio-level-01 (work in progress), March 2011.

[I-D.lennox-avt-srtp-encrypted-extension-headers]
Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-lennox-avt-srtp-encrypted-extension-headers-02 (work in progress), October 2010.

[I-D.perkins-avt-srtp-vbr-audio]
Perkins, C. and J. Valin, "Guidelines for the use of Variable Bit Rate Audio with Secure RTP", draft-perkins-avt-srtp-vbr-audio-05 (work in progress), December 2010.

[ITU.G711.1988]
International Telecommunications Union, "Pulse Code Modulation (PCM) of Voice Frequencies", ITU-T Recommendation G.711, November 1988.

[ITU.P56.1993]
International Telecommunications Union, "Objective Measurement of Active Speech Level", ITU-T Recommendation P.56, March 1988.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

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

Appendix A. Open issues

- In order to more accurately determine signal-to-noise ratio, it would be useful for a sender to also send its estimate of its current audio noise floor. If so, it's unclear whether this would be better as a separate header extension element, or added to this header extension element.
- It has been suggested to reference ITU P.56 [ITU.P56.1993] for level measurement. This needs to be investigated.

Appendix B. Changes From Earlier Versions

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

B.1. Changes From Draft -01

- Added references to the sample level calculator in [I-D.ietf-avtext-mixer-to-client-audio-level].
- Changed affiliation for Emil Ivov.

Authors' Addresses

Jonathan Lennox (editor)
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Emil Ivov
Jitsi
Strasbourg 67000
France

Email: emcho@jitsi.org

Enrico Marocco
Telecom Itialia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

E. Ivov, Ed.
Jitsi
E. Marocco, Ed.
Telecom Italia
J. Lennox
Vidyo, Inc.

March 14, 2011

A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication
draft-ietf-avtext-mixer-to-client-audio-level-01

Abstract

This document describes a mechanism for RTP-level mixers in audio conferences to deliver information about the audio level of individual participants. Such audio level indicators are transported in the same RTP packets as the audio data they pertain to.

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 September 15, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Framework for Conferencing with the Session Initiation Protocol (SIP) defined in RFC 4353 [RFC4353] presents an overall architecture for multi-party conferencing. Among others, the framework borrows from RTP [RFC3550] and extends the concept of a mixer entity "responsible for combining the media streams that make up a conference, and generating one or more output streams that are delivered to recipients". Every participant would hence receive, in a flat single stream, media originating from all the others.

Using such centralized mixer-based architectures simplifies support for conference calls on the client side since they would hardly differ from one-to-one conversations. However, the method also introduces a few limitations. The flat nature of the streams that a mixer would output and send to participants makes it difficult for users to identify the original source of what they are hearing.

Mechanisms that allow the mixer to send to participants cues on current speakers (e.g. the CSRC fields in RTP [RFC3550]) only work for speaking/silent binary indications. There are, however, a number of use cases where one would require more detailed information. Possible examples include the presence of background chat/noise/ music/typing, someone breathing noisily in their microphone, or other cases where identifying the source of the disturbance would make it easy to remove it (e.g. by sending a private IM to the concerned party asking them to mute their microphone). A more advanced scenario could involve an intense discussion between multiple participants that the user does not personally know. Audio level information would help better recognize the speakers by associating with them complex (but still human readable) characteristics like loudness and speed for example.

One way of presenting such information in a user friendly manner would be for a conferencing client to attach audio level indicators to the corresponding participant related components in the user interface as displayed in Figure 1.

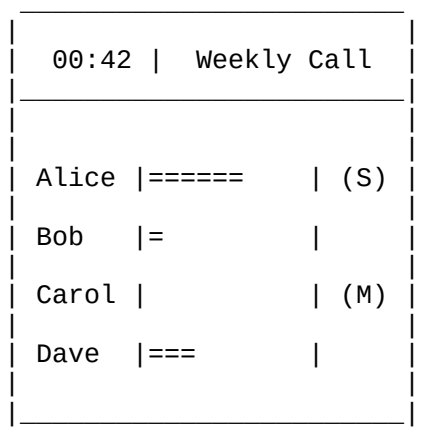

Figure 1: Displaying detailed speaker information to the user by including audio level for every participant.

Implementing a user interface like the above requires analysis of the media sent from other participants. In a conventional audio conference this is only possible for the mixer since all other conference participants are generally receiving a single, flat audio stream and have therefore no immediate way of determining individual audio levels.

This document specifies an RTP extension header that allows such mixers to deliver audio level information to conference participants by including it directly in the RTP packets transporting the corresponding audio data.

2. 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].

3. Protocol Operation

According to RFC 3550 [RFC3550] a mixer is expected to include in outgoing RTP packets a list of identifiers (CSRC IDs) indicating the sources that contributed to the resulting stream. The presence of such CSRC IDs allows RTP clients to determine, in a binary way, the active speaker(s) in any given moment. RTCP also provides a basic mechanism to map the CSRC IDs to user identities through the CNAME

field. More advanced mechanisms, may exist depending on the signaling protocol used to establish and control a conference. In the case of the Session Initiation Protocol [RFC3261] for example, the Event Package for Conference State [RFC4575] defines a <src-id> tag which binds CSRC IDs to media streams and SIP URIs.

This document describes an RTP header extension that allows mixers to indicate the audio-level of every conference participant (CSRC) in addition to simply indicating their on/off status. This new header extension is based on the "General Mechanism for RTP Header Extensions" [RFC5285].

Each instance of this header contains a list of one-octet audio levels expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov(see Section 4 and Section 5). Appendix A provides a reference implementation indicating one way of obtaining such values from raw audio samples.

Every audio level value pertains to the CSRC identifier located at the corresponding position in the CSRC list. In other words, the first value would indicate the audio level of the conference participant represented by the first CSRC identifier in that packet and so forth. The number and order of these values MUST therefore match the number and order of the CSRC IDs present in the same packet.

When encoding audio level information, a mixer SHOULD include in a packet information that corresponds to the audio data being transported in that same packet. It is important that these values follow the actual stream as closely as possible. Therefore a mixer SHOULD also calculate the values after the original contributing stream has undergone possible processing such as level normalization, and noise reduction for example.

Note that in some cases a mixer may be sending an RTP audio stream that only contains audio level information and no actual audio. Updating a (web) interface conference module may be one reason for this to happen.

It may sometimes happen that a conference involves more than a single mixer. In such cases each of the mixers MAY choose to relay the CSRC list and audio-level information they receive from peer mixers (as long as the total CSRC count remains below 16). Given that the maximum audio level is not precisely defined by this specification, it is likely that in such situations average audio levels would be perceptibly different for the participants located behind the different mixers.

4. Header Format

The audio level indicators are delivered to the receivers in-band using the "General Mechanism for RTP Header Extensions" [RFC5285]. The payload of this extension is an ordered sequence of 8-bit audio level indicators encoded as per Section 5.

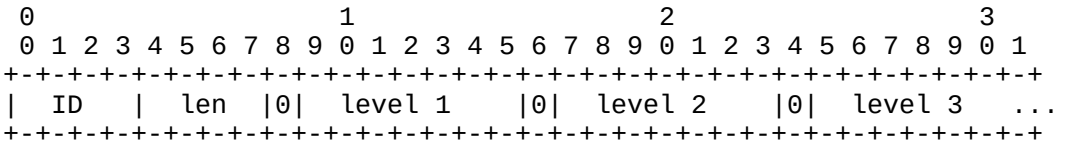

Figure 2: Audio level indicators extension format

The 4-bit len field is the number minus one of data bytes (i.e. audio level values) transported in this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows. A value of 15 is not allowed by this specification and it MUST NOT be used as the RTP header can carry a maximum of 15 CSRC IDs. The maximum value allowed is therefore 14 indicating a following sequence of 15 audio level values.

Note that use of the two-byte header defined in RFC 5285 [RFC5285] follows the same rules the only change being the length of the ID and len fields.

5. Audio level encoding

Audio level indicators are encoded in the same manner as audio noise level in the RTP Payload Comfort Noise specification [RFC3389] and audio level in the RTP Extension Header for Client-to-mixer Audio Level Notification [I-D.ietf-avtext-client-to-mixer-audio-level] specification. The magnitude of the audio level is packed into the least significant bits of one audio-level byte with the most significant bit unused and always set to 0 as shown below in Figure 3.

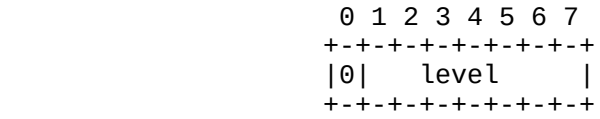

Figure 3: Audio Level Encoding

The audio level is expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov. dBov is the level, in decibels, relative to the overload point of the system, i.e. the maximum-amplitude signal that can be handled by the system without clipping. (Note: Representation relative to the overload point of a system is particularly useful for digital implementations, since one does not need to know the relative calibration of the analog circuitry.) For example, in the case of u-law (audio/pcmu) audio [ITU.G.711], the 0 dBov reference would be a square wave with values +/- 8031. (This translates to 6.18 dBm0, relative to u-law's dBm0 definition in Table 6 of G.711.)

To simplify implementation of the encoding procedures described here, this specification provides a sample Java implementation (Appendix A) demonstating one way it can be achieved.

6. Signaling Information

The URI for declaring the audio level header extension in an SDP extmap attribute and mapping it to a local extension header identifier is "urn:ietf:params:rtp-hdrext:csrc-audio-level". There is no additional setup information needed for this extension (i.e. no extensionattributes).

An example attribute line in the SDP, for a conference might be:

```
a=extmap:7 urn:ietf:params:rtp-hdrext:csrc-audio-level
```

The above mapping will most often be provided per media stream (in the media-level section(s) of SDP, i.e., after an "m=" line) or globally if there is more than one stream containing audio level indicators in a session.

Presence of the above attribute in the SDP description of a media stream indicates that some or all RTP packets in that stream would contain the audio level information RTP extension header.

Conferencing clients that support audio level indicators and have no mixing capabilities SHOULD always include the direction parameter in

the "extmap" attribute setting it to "recvonly". Conference focus entities with mixing capabilities MAY omit the direction or set it to "sendrecv" in SDP offers. Such entities SHOULD set it to "sendonly" in SDP answers to offers with a "recvonly" parameter and to "sendrecv" when answering other "sendrecv" offers.

The following Figure 4 and Figure 5 show two example offer/answer exchanges between a conferencing client and a focus, and between two conference focus entities.

```
v=0
o=alice 2890844526 2890844526 IN IP6 host.example.com
c=IN IP6 host.example.com
t=0 0
m=audio 49170 RTP/AVP 0 4
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=extmap:1/recvonly urn:ietf:params:rtp-hdrext:csrc-audio-level

v=0
i=A Seminar on the session description protocol
o=conf-focus 2890844730 2890844730 IN IP6 focus.example.net
c=IN IP6 focus.example.net
t=0 0
m=audio 52543 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendonly urn:ietf:params:rtp-hdrext:csrc-audio-level
```

A client-initiated example SDP offer/answer exchange negotiating an audio stream with one-way flow of of audio level information.

Figure 4

```
v=0
i=Un seminaire sur le protocole de description des sessions
o=fr-focus 2890844730 2890844730 IN IP6 focus.fr.example.net
c=IN IP6 focus.fr.example.net
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendrecv urn:ietf:params:rtp-hdrext:csrc-audio-level

v=0
i=A Seminar on the session description protocol
o=us-focus 2890844526 2890844526 IN IP6 focus.us.example.net
c=IN IP6 focus.us.example.net
t=0 0
m=audio 52543 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendrecv urn:ietf:params:rtp-hdrext:csrc-audio-level
```

An example SDP offer/answer exchange between two conference focus entities with mixing capabilities negotiating an audio stream with bidirectional flwo of audio level information.

Figure 5

7. Security Considerations

1. This document defines a means of attributing audio level to a particular participant in a conference. An attacker may try to modify the content of RTP packets in a way that would make audio activity from one participant appear as coming from another.
2. Furthermore, the fact that audio level values would not be protected even in an SRTP session may be of concern in some cases where the activity of a particular participant in a conference is confidential.
3. Both of the above are concerns that stem from the design of the RTP protocol itself and they would probably also apply when using CSRC identifiers the way they were specified in RFC 3550 [RFC3550]. It is therefore important that according to the needs of a particular scenario, implementors and deployers consider use of a lower level security and authentication mechanism.

8. IANA Considerations

This document defines a new extension URI that, if approved, would need to be added to the RTP Compact Header Extensions sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry, according

to the following data:

```
Extension URI: urn:ietf:params:rtp-hdrext:csrc-audio-level
Description:   Mixer-to-client audio level indicators
Contact:       emcho@jitsi.org
Reference:     RFC XXXX
```

9. Open Issues

At the time of writing of this document the authors have no clear view on how and if the following list of issues should be address here:

1. Audio levels in video streams. This specification allows use of audio level values in "silent" audio streams that don't otherwise carry any payload thus allowing their delivery within systems where the various focus/mixer components communicate with each other as conference participants. The same train of thought may very well justify audio level transport in video streams.
2. It has been suggested to reference ITU P.56 [ITU.P56.1993] for level measurement. This needs to be investigated.

10. Acknowledgments

Lyubomir Marinov contributed level measurement and rendering code.

Roni Even, Ingemar Johansson, Michael Ramalho and several others provided helpful feedback over the dispatch mailing list.

Jitsi's participation in this specification is funded by the NLnet Foundation.

11. Changes From Earlier Versions

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

11.1. Changes From Draft -00

- Added code for sound pressure calculation and measurement in "APPENDIX A. Reference Implementation".
- Changed affiliation for Emil Ivov.
- Removed "Appendix: Design choices".

12. References

12.1. 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.

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

12.2. Informative References

[I-D.ietf-avtext-client-to-mixer-audio-level] Lennox, J., Ivov, E., and E. Marocco, "A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", draft-ietf-avtext-client-to-mixer-audio-level-00 (work in progress), February 2011.

[ITU.G.711] International Telecommunications Union, "Pulse Code Modulation (PCM) of Voice Frequencies", ITU-T Recommendation G.711, November 1988.

[ITU.P56.1993] International Telecommunications Union, "Objective Measurement of Active Speech Level", ITU-T Recommendation P.56, March 1988.

[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.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353,

February 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

Appendix A. Reference Implementation

This appendix contains Java code for a reference implementation of the level calculation and rendering methods.The code is not normative and by no means the only possible implementation. Its purpose is to help implementors add audio level support to mixers and clients.

The Java code consists of the following files and methods:

AudioLevelCalculator.java: Calculates the sound pressure level of a signal with specific samples. Can be used in mixers to generate values suitable for the level extension headers.

AudioLevelRenderer.java: Helps adjust a sequence of pressure levels so that they would appear "natural" to users. Can be used by clients and applied over the values received in a level extension header so that displayed levels would change smoothly and correspond to user experience.

The implementation is provided in Java but does not rely on any of the language specific and can be easily ported to another.

A.1. AudioLevelCalculator.java

```
/**
 * Calculates the audio level of specific samples of a singal based on
 * sound pressure level.
 */
public class AudioLevelCalculator
{

    /**
     * Calculates the sound pressure level of a signal with specific
     * <tt>samples</tt>.
     *
     * @param samples the samples of the signal to calculate the sound
     * pressure level of. The samples are specified as an <tt>int</tt>
     * array starting at <tt>offset</tt>, extending <tt>length</tt>
     * number of elements and each <tt>int</tt> element in the specified
     * range representing a 16-bit sample.
     *
```

```
     * @param offset the offset in <tt>samples</tt> at which the samples
     * start
     * @param length the length of the signal specified in
     * <tt>samples<tt> starting at <tt>offset</tt>
     * @return the sound pressure level of the specified signal
     */
    public static int calculateSoundPressureLevel(
        int[] samples, int offset, int length)
    {
        /*
         * Calcuate the root mean square of the signal i.e. the
         * effective sound pressure.
         */
        double rms = 0;

        for (; offset < length; offset++)
        {
            double sample = samples[offset];

            sample /= Short.MAX_VALUE;
            rms += sample * sample;
        }
        rms = (length == 0) ? 0 : Math.sqrt(rms / length);

        /*
         * The sound pressure level is a logarithmic measure of the
         * effectivesound pressure of a sound relative to a reference
         * value and is measured in decibels.
         */
        double db;

        /*
         * The minimum sound pressure level which matches the maximum
         * of the sound meter.
         */
        final double MIN_SOUND_PRESSURE_LEVEL = 0;
        /*
         * The maximum sound pressure level which matches the maximum
         * of the sound meter.
         */
        final double MAX_SOUND_PRESSURE_LEVEL
            = 127 /* HUMAN TINNITUS (RINGING IN THE EARS) BEGINS */;

        if (rms > 0)
        {
            /*
             * The commonly used "zero" reference sound pressure in air
             * is 20 uPa RMS, which is usually considered the threshold
```

```
             * of human hearing.
             */
            final double REF_SOUND_PRESSURE = 0.00002;

            db = 20 * Math.log10(rms / REF_SOUND_PRESSURE);

            /*
             * Ensure that the calculated level is within the minimum
             * and maximum sound pressure level.
             */
            if (db < MIN_SOUND_PRESSURE_LEVEL)
                db = MIN_SOUND_PRESSURE_LEVEL;
            else if (db > MAX_SOUND_PRESSURE_LEVEL)
                db = MAX_SOUND_PRESSURE_LEVEL;
        }
        else
        {
            db = MIN_SOUND_PRESSURE_LEVEL;
        }

        return (int) db;
    }
}
```

AudioLevelCalculator.java

A.2. AudioLevelRenderer.java

```
/**
 * Helps adjust a sequence of pressure levels so that they would appear
 * "natural" to users. Can be used by clients and applied over the
 * values received in a level extension header so that displayed levels
 * would change smoothly and correspond to user experience..
 */
public class AudioLevelRenderer
{
    /**
     * The last audio level displayed by
     * {@link AudioLevelCalculator#displayAudioLevel(int, int, int)}.
     */
    private int lastAudioLevel = 0;

    /**
     * Returns a specific sound pressure level as an animated (i.e.
     * does not jump up and down too much in a single update) audio
     * level.
```

```
     *
     * @param spl the sound pressure level to be displayed
     * @param minAudioLevel the minimum of the UI range which is used
     * to depict audio levels
     * @param maxAudioLevel the maximum of the UI range which is used
     * to depict audio levels
     * @return a sound pressure level that can be displayed to the user.
     */
    public int renderAudioLevel(
            int spl, int minAudioLevel, int maxAudioLevel)
    {
        /*
         * The minimum sound pressure level that the UI is interested in
         * displaying.
         */
        final double MIN_SPL_TO_DISPLAY = 40 /* A WHISPER */;

        /*
         * The maximum sound pressure level that the UI is interested in
         * displaying.
         */
        final double MAX_SPL_TO_DISPLAY = 85 /* HEARING DAMAGE */;

        int audioLevel;

        if (spl < MIN_SPL_TO_DISPLAY)
            audioLevel = minAudioLevel;
        else if (spl > MAX_SPL_TO_DISPLAY)
            audioLevel = maxAudioLevel;
        else
        {
            /*
             * Depict the range between "A WHISPER" and the beginning of
             * "HEARING DAMAGE".
             */
            audioLevel
                = (int)
                    (((spl - MIN_SPL_TO_DISPLAY)
                            / (MAX_SPL_TO_DISPLAY - MIN_SPL_TO_DISPLAY))
                        * (maxAudioLevel - minAudioLevel));
            if (audioLevel < minAudioLevel)
                audioLevel = minAudioLevel;
            else if (audioLevel > maxAudioLevel)
                audioLevel = maxAudioLevel;
        }

        /*
         * Animate the audio level so that it does not jump up and down
```

```
         * too fast.
         */
        lastAudioLevel
                = (int) (lastAudioLevel * 0.8 + audioLevel * 0.2);

        /* Return the displayable audio level. */
        return lastAudioLevel;
    }
}
```

AudioLevelRenderer.java

Authors' Addresses

Emil Ivov (editor)
Jitsi
Strasbourg 67000
France

Email: emcho@jitsi.org

Enrico Marocco (editor)
Telecom Italia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

R. Blom
Y. Cheng
F. Lindholm
J. Mattsson
M. Naslund
K. Norrman
Ericsson

March 14, 2011

SRTP Store-and-Forward Use Cases and Requirements
draft-mattsson-srtp-store-and-forward-04

Abstract

The Secure Real-time Transport Protocol (SRTP) was designed to allow simple and efficient protection of RTP. To provide this, encryption and authentication of media and control signaling are tightly coupled to the RTP session, and the information in the RTP header. Hence, in general, it is not possible to perform store-and-forward of protected media using SRTP.

This document gives, based on a use case analysis, requirements that SRTP and new SRTP transforms need to satisfy in order to allow secure store-and-forward operation. A first outline on how to introduce the needed new functionality and transforms in SRTP is also presented.

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 September 15, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Secure Real-time Transport Protocol (SRTP) [RFC3711] is a profile of the Real-time Transport Protocol (RTP) [RFC3550], and it provides confidentiality, message authentication, and replay protection to both RTP and RTCP (Real-time Transport Control Protocol).

SRTP was designed to protect real-time point-to-point communications and is, as presently defined, not aimed for communication solutions that include non-trusted store-and-forward middleboxes, i.e. middleboxes that should not have access to cleartext media, but still should have access to other data in order to retransmit media according to RTP standard procedures.

Media in need of end-to-end (e2e) protection could e.g. be real-time voice and video information/media clips for internal use by personnel in enterprises or authorities. There are also multimedia telephony applications utilizing media mailboxes and other store-and-forward functions that need e2e protection. Protection e2e could also be needed to protect subscribed media like commercial-free radio and television that is distributed over the Internet.

A typical use case is store-and-forward media distributions systems. Many of those systems require that media is confidentiality protected e2e between the media source and the media rendering device; this to prevent illegitimate media intercept or sharing. At the same time the communication should be hop-by-hop (hbh) protected to prevent malicious users from performing denial of service attacks by sending bogus data to store-and-forward middleboxes. Methods like the Packet-switched Streaming Service (PSS) [3GPP.26.234] exhibit the properties needed for secure store-and-forward operation, but they are part of larger frameworks tailored for very specific use cases. Thus, it would be desirable to be able to offer use of SRTP as a general lightweight mechanism to achieve this type of protection.

Trying to use SRTP with store-and-forward middleboxes reveals two main problems:

The first problem is due to the fact that the incoming and outgoing RTP streams in general are independent; received RTP packets cannot just be stored and later retransmitted. This in particular implies that SRTP with currently defined transforms cannot be applied. For details, see Section 3.

It should be noted that store-and-forward of media in most cases requires that side information is available when retransmitting received media. Such side information, e.g. RTP timestamp information, may come from the RTP header, RTCP messages, and session

definition data.

The second problem is due to the fact that 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. To resolve these problems, SRTP needs extensions that in an efficient and coherent way support store-and-forward use cases.

The objective of this document is to explore use cases for a SRTP store-and-forward solution, derive associated requirements, present, and discuss an approach for a solution.

2. 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 [RFC2119].

Definitions of terms and notation will, unless otherwise indicated, be as defined in [RFC3711].

- The term authentication will be used to denote message authentication and message integrity protection.
- By RTP transport protection or simply transport protection, we mean protection (confidentiality, authentication, etc.) of streamed RTP packets. This is provided by SRTP according to [RFC3711].
- By media protection, we similarly mean e2e protection of the application payloads carried in RTP. SRTP provides media protection, but only during transport (see above). A (protected) media stream similarly refers to (protected) media payloads streamed using RTP.
- A store-and-forward e2e session is defined as the set of store-and-forward e2e protected data produced under a single so called e2e (cryptographic) context. A store-and-forward e2e session may comprise several so called store-and-forward sources, i.e. several distinct logical e2e media streams to be protected by the same e2e context.
- A store-and-forward hbh session is defined as the set of store-and-forward hbh protected data produced under a single so called hbh context.

3. Selected SRTP Background Facts

SRTP as currently specified has the properties described below, which explain why it cannot be directly used in store-and-forward applications. The description also indicates how a SRTP store-and-forward solution could be designed.

- All current SRTP transforms use the RTP header as input. AES-CTR uses the SSRC and the packet index to calculate the IV (Initialization Vector), AES-f8 uses even more header parameters, and HMAC-SHA1 authenticates the full RTP header. The SSRC is typically determined by the key management protocol and the packet index includes the RTP sequence number, which should be randomly chosen according to RTP [RFC3550]. All this means that there are no standard compliant ways to receive SRTP protected packets in one stream and later just retransmit the packets as they were received.

- Even if the SRTP relevant RTP parameters like SSRC and the SRTP index could be determined beforehand for the retransmission stream, it would not allow a client to randomly seek in a stream without renegotiating the session, as it would lead to misalignment between the packet index used for streaming and the packet index used by SRTP at the originator. If the user jumps to a different part of the stream, it is impossible to continue increasing the RTP sequence number stepwise while at the same time keeping it equal to the sequence number needed for decryption. Jumping backward (e.g. media rewind) would cause even more problems as the retransmitted packets would be discarded by the SRTP replay protection.

- The encryption key and the authentication key are both derived from the same master key in SRTP, see Figure 1. This means that a client which is able to derive e.g. the authentication key will also always have access to the encryption key making it impossible to use say the session encr_key for e2e protection and the session auth_key for hbh protection.

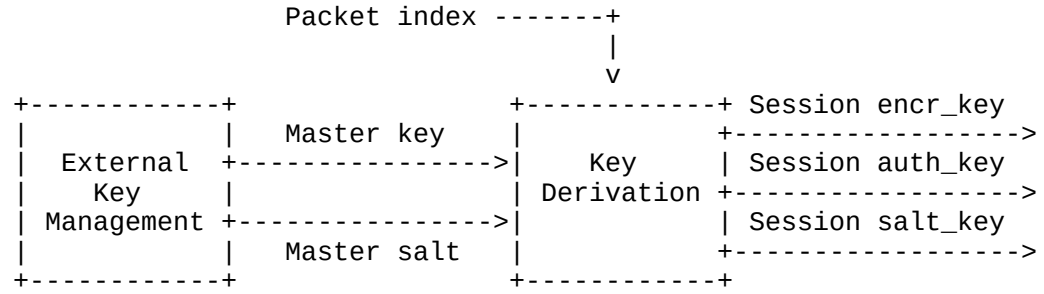

Figure 1: SRTP key derivation

4. Use Cases

The use cases below were chosen to illustrate media streaming scenarios where the current SRTP specification [RFC3711] does not provide sufficient functionality. These use cases provide context and general rationale for the requirements presented in Section 5.

Note that the necessary key distribution and media session setup is out of scope for this document, and will thus not be discussed in any detail in the use cases below. However, as key management is an integral part of a complete store-and-forward solution, some approaches to the necessary key distribution and media session setup for some of the use cases are discussed in Appendix A.

4.1. Trust Model and Assumptions

The trust model assumed in this document includes two parties who wish to communicate securely via one or more honest but curious middleboxes. This means that the communicating parties trust the middlebox to deliver the media as expected, but they do not trust it with cleartext data. In the use cases below, there is no example of multiple (sequential) middleboxes, but it is a natural generalization and it seems warranted to cover this case as well.

4.2. Media Distribution Use Cases

4.2.1. Streaming Pre-encrypted Media

A content provider wants to distribute high value media to clients. The content provider distributes the media via a streaming server that should not have access to cleartext media, typically because the content provider does not trust it. In one scenario, the content provider streams the media to the streaming server where the media is stored in a protected format. In another scenario, the protected

media may be delivered to the streaming server via e.g. file transfer. These use cases correspond to the use of pre-encryption in media distribution. In both cases, protected media is available in the streaming server for later transmission to different clients.

Even in cases when the streaming server could be trusted with cleartext data there are reasons why one would like to avoid performing encryption in the streaming server itself. One reason is to use pre-encryption to offload the streaming server the task of encrypting the media. If the media is pre-encrypted, the streaming server only needs to add integrity protection (for hbh protection) to the encrypted media before streaming it to the clients. Clients are trusted by the content provider and have access to the encryption key. When a client receives a packet, the authenticity is checked using a security context shared with the streaming server and the decryption is performed using a security context shared with the content provider.

4.2.2. Video on Demand

Some protected media is offered as video on demand where users can watch selected video clips at any time. The media is unicasted and the clients are offered random seek functionality which allow them to quickly jump to any part of the video. Other features offered may be rendering with speed translation as in fast forward and slow motion rendering. These features can be used to skip parts of the video or jump backward to see interesting parts again. The problem here is jumping back and forth and performing rendering speed translations in an e2e protected media stream with associated implications on synchronization and interactions with replay protection.

4.2.3. Caching Protected Media in the Network

High value encrypted media (e.g. Internet Protocol Television (IPTV), and radio) is broadcasted in a network. Only clients trusted by the content provider have access to the encryption key. A network node 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.

4.2.4. Recording Encrypted Media at Home

High value encrypted media (e.g. IPTV, and radio) is broadcasted in a network. Only clients trusted by the content provider have access to the encryption key. A user is recording the media on a HDD (Hard Disk Drive), but does not yet have a license, or have a license that does not allow cleartext copying. The media is therefore stored in protected format on the HDD. There is however, a strong need for the HDD to be able to check the integrity of the media before it is stored. Otherwise, a DoS attack may fill the HDD with garbage.

4.3. Answering Machine Use Cases

4.3.1. Storing/Caching Encrypted Media

Operators commonly provide an answering machine service to their customers. In this case, the communicating parties (the caller and the callee) may not wish to disclose the media to any other party, and hence want to apply encryption between each other. This requires that they are able to establish a shared key. The answering machine acts as a store-and-forward middlebox, which stores encrypted data and retransmits it to the callee. The answering machine may act as a streaming server when sending the data to the callee, and will then not use the exact same RTP headers on the outgoing SRTP traffic as was used on the incoming SRTP traffic. SRTP as specified in [RFC3711] will not work in this case, since parts of the RTP header are input to the encryption/authentication transforms.

An alternative forwarding of the recorded media from the answering machine to the callee could be by file transfer, e.g. sending the recorded media in the format that was used to store it. Such forwarding would not be according to SRTP, but would still yield end-to-end protection of the media. Note however, that decryption and rendering would be similar to part of an enhanced SRTP solution.

4.3.2. Transport Protection

To avoid that the answering machine is filled up with bogus data, it is necessary for the answering machine to authenticate the sender of the traffic, and further, to verify the authenticity of the incoming traffic. This poses a problem for SRTP as of [RFC3711] in that the message authentication requires a session key shared with the answering machine, but the encryption key shall as discussed above not be available to it. This implies that there is a need for two independent security contexts, one end-to-end and one hop-by-hop.

When the callee retrieves the media from the answering machine, message authentication is also beneficial. There are two

possibilities. Since the answering machine is trusted to maintain and redistribute the media, it may be sufficient to provide message authentication between the answering machine and the callee. In addition, here it would be necessary to have a separation between the e2e protection and the hbh protection. A second option is that authentication is applied from the caller to the callee. However, if the authentication is applied in that way, the answering machine will not be able to verify the integrity of the incoming traffic from the caller. It is of course also possible that message authentication is desired for any combination of endpoints, i.e. between the caller and the callee, between the caller and the answering machine, and between the answering machine and the callee.

4.3.3. Playback of Media Stream

When a user listens to the messages stored on the answering machine, it is useful to be able to rewind and/or fast forward in the media stream. For SRTP as of [RFC3711], this is not possible. The reason for that is that even if the same payloads can be reinserted in the stream by the answering machine, the RTP sequence number is steadily increasing on a per packet basis. Since the synchronization of the encryption transforms is based on the RTP sequence number, the decryption will fail. In addition, message authentication will fail since the authentication according to [RFC3711] shall cover the header of the RTP packet. This implies that the payload and the media have to be protected by a mechanism that is independent of parameters used in the transport protocol.

4.3.4. Multiple Callers

Several messages may be left on the answering machine, received in different sessions and possibly from different callers. The result of this is that different contexts (keys) were used to encrypt the media. Depending on how the callee retrieves the messages from the answering machine, different options are possible. One option is to retrieve each message as a separate stream, and in this case, a separate session is required per message. Another option is to somehow switch security contexts within an ongoing hbh session.

4.4. Centralized Conferencing Use Case

Another use case is a conference bridge that either is not to be trusted with the cleartext media or do not have the processing power to decrypt and re-encrypt the media from a large number of participants. In this case, the conference bridge cannot act as a mixer, but in some cases, that may be a reasonable assumption. In this setting, the media may be repackaged by the conferencing server into RTP packets with different headers compared to the incoming

traffic. As described in Section 3, this causes authentication and decryption to fail in SRTP. An example is Push-To-Talk solutions, where only one user at a time is allowed to talk. Another example where this is especially interesting are video conferencing applications, were a conference server does not work as a media mixer, but rather as hub for the conference participants. In such a setup, the application of group based approaches for security may be desirable for the e2e protection of media.

5. Requirements

The use cases above show that to enable store-and-forward in an extended SRTP, it has to in an efficient way support the following requirements:

- 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.

  The requirements imply that the media protection format has to include a SRTP SaF Source (SSS) field for robust operation. The SSS can be thought of as an "e2e SSRC".

- 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. This implies a Packet Unique Value (PUV) used on e2e basis in order for the receiver to identify a media payload's position within the overall media stream.

- 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.

- 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.

- Change of transport independent media protection security context

  It MUST be possible to signal to the receiver the current media protection security context to use. It MUST be possible to change the e2e security context within an ongoing hbh session.

  This is needed to allow single stream multiplexing of e.g. protected media "clips" which were generated using different transport independent media protection security contexts

  The requirements imply that the media protection format has to include a Crypto Context Indicator (CCI) field for robust operation. The CCI can be thought of as a generalized MKI and may be defined to also include all the MKI based functionality defined in [RFC3711].

6. Solution Outline

In this section, a first outline on how to introduce the needed new functionality and transforms in SRTP is presented. For a complete description, including a packet format specification and a detailed transform description, see [I-D.naslund-srtp-saf].

6.1. Overview

The stated requirements above seem possible to meet by implementing a few minor additions to SRTP. These additions mainly address new SRTP transforms, introduction of media and transport protection crypto context definitions, together with key handling and key derivation.

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 could either be done with an Authenticated Encryption (AE) transform, or with separate encryption and authentication transforms. The media protection should rely on two explicit values for cryptographic synchronization, the Packet Unique Value (PUV) and the SRTP SaF Source (SSS), which are 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 is denoted hbh key.

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 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 new SRTP functionality with combined media and transport transform and handling of two security contexts. In the following, we will denote such a combined transform a Compound Transform (CT). The store-and-forward middlebox can rely solely on [RFC3711], using already existing functionality for store-

and-forward operation, given that the transport transform in the compound transform is equivalent to a transform defined for [RFC3711]. However, there are some practical reasons why also the middlebox needs to have some "knowledge" of the e2e part of the protection, see below.

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 store-and-forward middlebox should be able to reuse a [RFC3711] compliant implementation of SRTP to first receive and then resend the media. However, there are some practical reasons why also the middlebox needs to have some "knowledge" of the e2e part of the protection, see below.

For RTCP the solution principles described for RTP applies. However, the main application for RTCP is to control the traffic over one hop, 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.

6.2. SRTP Store-and-Forward 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.

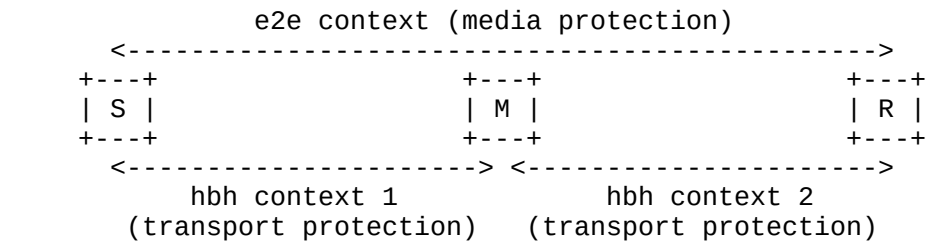

Figure 2: Context sharing (Sender, Middlebox, Receiver)

The SRTP cryptographic context concept is reusable for the proposed solution. Conceptually, the originator and the intended end-receiver 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.

To comply with the trust model of the use cases above, 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 should be as defined in [RFC3711], while the used e2e context is either implicitly identified in the session setup or its identification relies on the proposed crypto context indicator (CCI).

A sender will use two cryptographic contexts: an e2e context used for payload protection to the end-receiver, and a hbh context used to secure the SRTP transport to the (first) intermediary. Similarly, the end-receiver 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 as required in Section 5, 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].

If several senders' payloads are multiplexed within the same stream from a server to a receiver (as discussed in Section 4.3.4) the receiver may need to switch between e2e contexts within an ongoing hbh session. This can be implemented using a mechanism similar to the SRTP MKI field in the e2e context (what is referred to as CCI above). The hbh context would, however, not need any change but could rely on an MKI field according to the current definition in [RFC3711].

6.3. Store-and-Forward 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.

With fields and processing as defined above, the SRTP store-and-forward packet format should look approximately like Figure 3

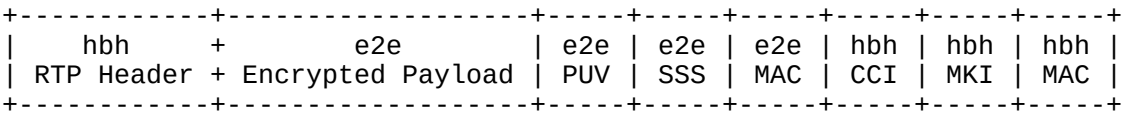

Figure 3: SRTP store-and-forward packet format

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

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

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

- MAC (e2e): This field is used to carry payload authentication data e2e.

- CCI: Crypto Context Identifier is used to signal hbh, which e2e cryptographic context to use.

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

6.4. 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]. As mentioned, 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.

7. Commented Example Usage

In this example use case, it is assumed that a single sender S wants to send a single e2e protected media stream to a receiver R. We make the natural (and necessary) assumption that the sender is made aware (e.g. by session setup signaling) that the media will be delivered/ stored in a middlebox M. Similarly, we assume the middlebox is aware that it is acting as a middlebox.

We assume the crypto contexts are defined to provide

- Integrity and confidentiality e2e (the media part)

- Integrity hbh (the transport part)

Clearly, other combinations are also possible. Any of the 15 possible (non-trivial) combinations of the security services confidentiality and integrity for the hbh and the e2e part could be specified for use. However, we feel that integrity and confidentiality on e2e basis combined with hbh integrity will be sufficient in most cases.

How the crypto contexts are setup (which key management protocol to use etc.) is out of scope. Still, it can be noted that in principle it could be done by having e.g. two MIKEY [RFC3830] exchanges, one between S and M and one between S and R.

1. S defines an e2e crypto context and forwards it to R. The e2e protection is configured to use both integrity and confidentiality protection. Note that for store-and-forward operation, the e2e crypto context has to be decided unilaterally by the sender.

2. S sets up an SRTP session with M, to have data forwarded to R; an hbh crypto context is agreed between them. The hbh context defines transport authentication and NULL transport encryption, which corresponds to transforms defined for [RFC3711].

3. S starts to transmit SRTP towards M, in effect using k_e2e for e2e media protection and k_hbh for hbh transport authentication.

4. Since M is aware of its role as a (receiving) middlebox, M configures itself to verify integrity but not to decrypt the payload. M stores the (protected) payloads together with relevant side information to be used when the media is forwarded. Note that M would perform exactly the same operations when storing unprotected media for later forwarding.

5. Later, R sets up a session with M to render the stored media. As R contacts a middlebox, an hbh crypto context, independent of the previous contexts, is agreed between R and M. In the reply, M includes the e2e context that was received from S.

6. Since M is aware of its role as a (sending) middlebox, the middlebox configures itself to not encrypt the payloads but only to add hbh transport authentication. M then transmits the authenticated media stream to R.

7. When receiving the SRTP packets from M, R first verifies the hbh transport authentication and then checks e2e media authentication and decrypts the payloads to retrieve the plaintext media.

8. Implications on SRTP

As the SRTP specification allows new transforms, the new transforms can be added with only minor implications.

The handling of dual security contexts (in the endpoints) is however a new feature, which will have to be introduced in SRTP.

The Key Derivation Function defined in [RFC3711] can be reused for both the e2e and the hbh security contexts.

9. Security Considerations

9.1. Media protection Transform

Any fixed keystream output, generated from the same inputs (i.e. key and IV) MUST only be used to encrypt once. Reusing such a key-stream (commonly called a "two-time pad") would almost certainly compromise security.

The new e2e transform accomplish packet-uniqueness by inclusion of the PUV and stream-uniqueness by inclusion of the SSS in the IV formation. Thus, the SSS MUST be unique among all the RTP streams within the same RTP session that share the same e2e master key. Master keys MAY be shared between streams belonging to the same RTP session, but it is RECOMMENDED that each stream have its own master key.

With the above conditions fulfilled, the security level of the media protection transform will equal the level offered by [RFC3711].

9.2. Replay Protection

Replay protection is only provided on hbh basis. Note that the requirements on random seek in the media stream rules out any general replay protection mechanism applied on an e2e basis, and that this threat falls outside the assumed trust model. Still, the PUV used offers possibility to implement application specific replay protection mechanisms.

10. Acknowledgements

The authors would like to thank Daniel Catrein, Steffen Fries, Frank Hartung, and Magnus Westerlund for their support and valuable comments.

11. 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.

12. References

12.1. Normative References

[I-D.naslund-srtp-saf] Blom, R., Cheng, Y., Lindholm, F., Mattsson, J., Naslund, M., and K. Norrman, "The Use of the Secure Real-time Transport Protocol (SRTP) in Store-and-Forward Applications", draft-naslund-srtp-saf-03 (work in progress), October 2009.

[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.

12.2. Informative References

[3GPP.26.234] 3GPP, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs", 3GPP TS 26.234 8.3.0, June 2009.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

Appendix A. Key Management

This informative appendix discusses possible ways to establish SRTP cryptographic contexts for store-and-forward scenarios. As described above there are two cryptographic contexts, i.e., an e2e context and an hbh context, and they should be independent of each other.

An hbh context is identified by the triplet <SSRC, destination IP address, destination port number> as defined in [RFC3711]. All currently available key management protocols that support SRTP, e.g. MIKEY, SDES, and DTLS-SRTP, can be used between sender/receiver and middlebox or between two middleboxes for negotiating hbh master keys and other security parameters.

The e2e context must also be identified and the identifier can be any transport independent value that uniquely determines the cryptographic context between a sender and a receiver. For instance, the sender could assign a unique id to the content to be transmitted and use such a Content ID (CID) to identify the e2e context. The CID is then sent to the middlebox at session setup time, and the CID and the e2e context are sent to the receiver at any time before the receiver is to render the media. Note that the CID discussed here is not the same as the proposed CCI. The CCI may be thought of as a mutant, short, in-band alias for the CID and is only used on hbh basis. The mapping between CID and CCI is then sent out-of-band for each hop, e.g. at session set-up for the respective hop. The receiver can thus (eventually) map the CCI received in SRTP packets to the correct CID and retrieve the corresponding e2e cryptographic context.

Therefore, for the e2e context additional information, i.e. CID and (CID, CCI)-mapping, needs to be transmitted, along with the key management protocol messages. Below we give two examples, addressing media distribution and answering machine use cases respectively. In the examples we use MIKEY over SIP/RTSP, but other key management protocols that support SRTP can also be used.

A.1. Key Management Example for Media Distribution

An example of session setup sequence for a media distribution use case (e.g. Video on demand) is shown in Figure 4. An end user (R) sends a SIP INVITE to the media service (S) to request the delivery of certain content. S replies with a 200 OK message, which includes the CID and a MIKEY message containing e2e master key and other parameters. In case of pre-encrypted content, the e2e context is the same for all users that are authorized to play the content.

The pre-encrypted content is stored in the streaming server (M).

When the end user wants to play the content, R sends an RTSP DESCRIBE message to M in order to obtain session description. M replies with 200 OK, carrying a MIKEY message for setting up the hbh context between M and R.

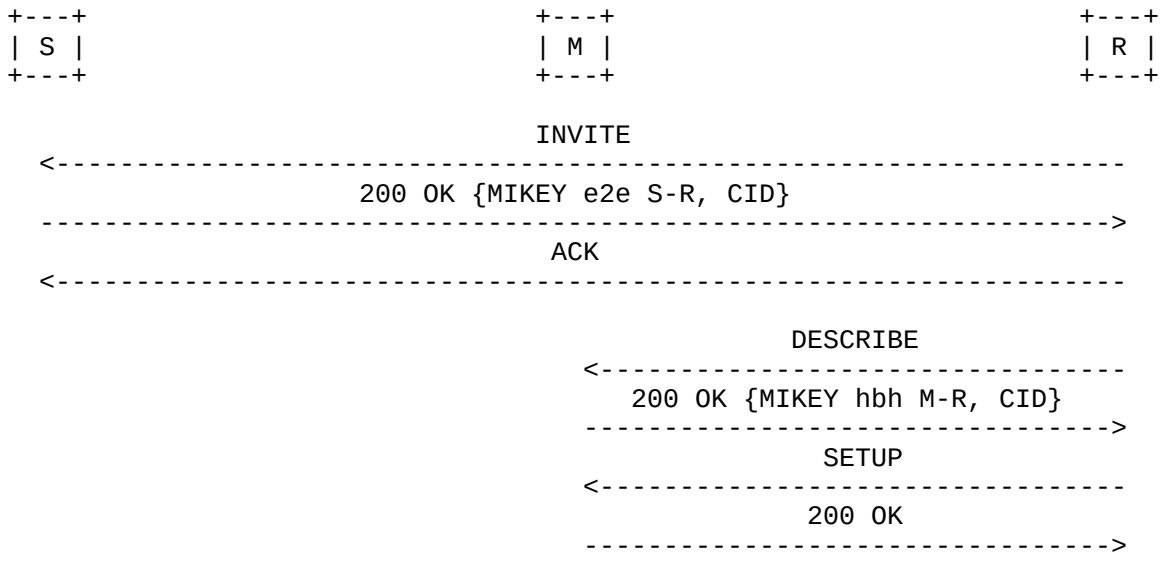

Figure 4: Session setup sequence for media distribution

A.2. Key Management Example for Answering Machine

Typically, a caller (S1) tries to reach the intended callee (R) directly. If R is not online, S1 is notified and redirected to an answering machine (M). S1 then knows it should run SRTP SaF. To signal that, S1 sends an INVITE with two MIKEY messages, one for setting up the e2e context between S1 and R, and the other for the hbh context between S1 and M. M cannot process the first MIKEY message but stores it. By processing the second MIKEY message, M agrees the hbh context with S1.

Another caller (S2) also wants to talk to R. Similarly, a hbh context is established between S2 and M, and M stores the e2e MIKEY message from S2 that is intended for R.

Later when R gets online and tries to retrieve stored data from M, R sends an INVITE to M and negotiates the hbh context between them. In the reply, M includes the two MIKEY messages carrying the e2e contexts that were received from S1 and S2 respectively, and adds the mappings between contexts and CCIs. A session setup sequence is shown in Figure 5.

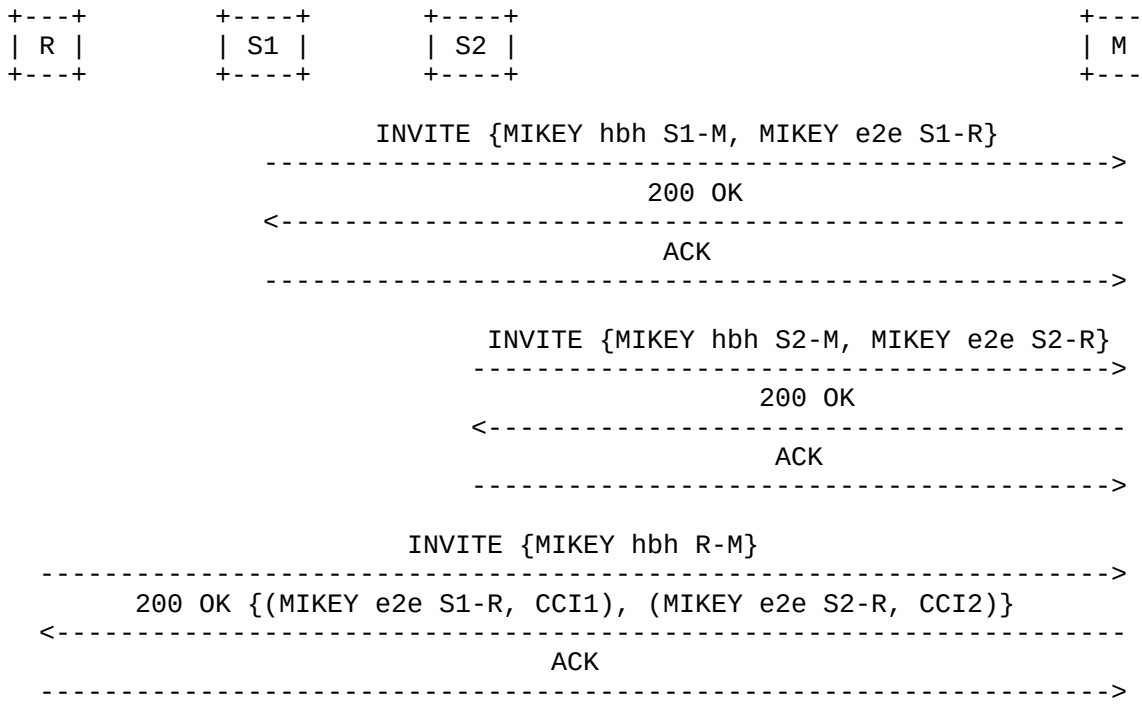

Figure 5: Session setup sequence for answering machine

Authors' Addresses

Rolf Blom
F. Lindholm
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 31 707
Email: rolf.j.blom@ericsson.com

Yi Cheng
F. Lindholm
SE-164 80 Stockholm
Sweden

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

Fredrik Lindholm
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 31 705
Email: fredrik.lindholm@ericsson.com

John Mattsson
Ericsson
SE-164 80 Stockholm
Sweden

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

Mats Naslund
Ericsson
SE-164 80 Stockholm
Sweden

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

Karl Norrman
Ericsson
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 44 502
Email: karl.norrman@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2011

R. Blom
Y. Cheng
F. Lindholm
J. Mattsson
M. Naslund
K. Norrman
Ericsson

March 14, 2011

The Use of the Secure Real-time Transport Protocol (SRTP) in Store-and-Forward Applications
draft-naslund-srtp-saf-04

Abstract

This memo describes the use of so called store-and-forward cryptographic transforms within the Secure Real-time Transport Protocol (SRTP). The motivation is to support use cases when two end-points communicate via one (or more) store-and-forward middleboxes that are not fully trusted to access the media content. One of the main aspects of the transform is to make the confidentiality and message authentication independent of the RTP header. Another central aspect is to enable identification of the cryptographic context (keys etc.). Besides the security of the end-points, also trust assumptions regarding the store-and-forward middleboxes 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 September 15, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

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] solves real-time end-to-end use cases, and does not consider use cases requiring Store-and-Forward (SaF) middleboxes. Such use cases are characterized by the need for a sender to deliver media to a receiver via a SaF middlebox. A SaF middlebox temporarily stores media and retransmits it to the intended receiver. Retransmission can be almost immediate (e.g. a push-to-talk group server), or be done at a much later time (e.g. a VoIP answering machine). The SaF middlebox is typically considered as semi-trusted, meaning that a SaF middlebox will store and deliver media as requested, but it cannot be excluded that a SaF middlebox will also try to extract the information (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 a so called store-and-forward extension 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 SaF 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 SaF middlebox. Indeed, the receiver may not even be on-line at the time when the source initiates the communication. Another part of the cryptographic context identifier is the SSRC, which may be modified by SaF 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 SaF 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 the store-and-forward use case and its associated trust model. The SaF use case and trust models is defined in Section 3. No claims are made about supporting also other use cases, though of course, all the original uses cases from [RFC3711] can also be supported.

The SaF use case implies a different trust model than that originally considered when designing SRTP. This manifests itself in terms of the need to ensure 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. Requirements on key management can be found in Sections 3.3 and 5.5.

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].

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

1.2.1. Notation and Definitions

DoS: Denial of service

e2e: end-to-end

hbh: hop-by-hop

SaF: Store-and-Forward

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.

2. SRTP

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). Note that the term "SRTP" may often be used to indicate SRTCP as well. SRTP is a framework that allows new security functions and new transforms to be added. In the sequel, we assume that the reader is familiar with the SRTP specification [RFC3711], its packet structure, and its processing rules.

This specification defines a so called Store-and-Forward extension to SRTP to permit communication via semi-trusted SaF middleboxes. As mentioned, the SRTP extensions defined herein are very similar in nature to the SRTP extensions previously defined in [RFC4383] to add source origin authentication support to SRTP. In both cases, the extensions needed are: definition of new cryptographic transforms, a new packet format including additional in-band context signaling, and extensions to the SRTP cryptographic context concept.

3. The Store-and-Forward Use Cases

3.1. Problem Statement

We consider RTP communication solutions that include semi-trusted SaF 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, SaF Middlebox, and Receiver. For each use case, we comment on aspects of the trust-model defined above.

Streaming Pre-encrypted Media: A content creator (S) distributes high value, encrypted content to clients (R). Distribution is made via a streaming server (M). From the content creator's point of view it is important that decrypted (plaintext) content is only made available to authorized clients. This means that S should be able to use a streaming server M, to which it assigns a bare minimum of sender-semi-trust. The clients may typically have some basic privacy requirement related to what type of content they access, but may otherwise be less concerned with whom else that also gets access to the content.

Recording Encrypted Media: Encrypted IPTV is broadcasted in a network. Only clients trusted by the content creator (S) should have access. Before having acquired a license to view the content, a user (R) records media on a Hard Disk Drive (M), where the media is stored in encrypted format, awaiting a license for rendering. Here, the trust in the HDD (M) by S and R is probably very asymmetric since the end-user most likely has a very strong trust in his personal home equipment. An additional requirement is the possibility for M to authenticate the data source S in order not to exhaust storage capacity with garbage.

Answering Machine: Operators commonly provide an answering machine service to their customers. Communicating parties (S and R) may not wish to disclose the media to any other party. Thus, the answering machine (M) acts as a SaF middlebox, which has to store encrypted data and retransmit it to the callee. In this use case, sender and receiver-semi-trust in M is likely to be fairly symmetric. In this case, it may be convenient to be able to splice several e2e protected streams (i.e. messages left by several different senders) into the same hbh communication session between M and R.

Further examples and more details can be found in Appendix A.

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 SaF middleboxes, which the SaF middleboxes then would store, eventually exhausting their storage space and/or corrupting the data stored.

3.2. Trust Model and Security Requirements

The following figure 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.

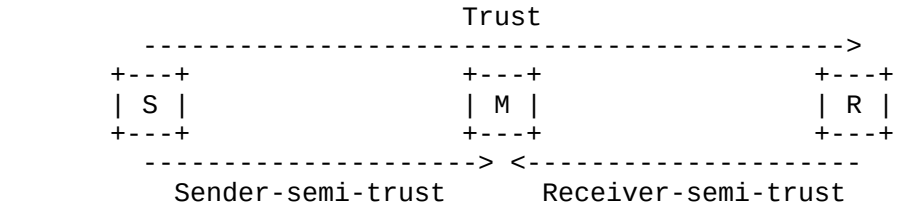

Figure 1: Trust Model (Sender, SaF Middlebox, Receiver)

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 SaF use cases. 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, depending on the application, the security requirements, and the processing/storage capacity, for some SaF middleboxes it will be REQUIRED to be able to authenticate the source before accepting to store/forward data. If this was not the case, some anonymous malicious party might exhaust the storage resources of M. Similarly, a more robust implementation of M should have means to authenticate also R in order to avoid wasting resources, responding to spoofed requests. That is, 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 SaF services provided by M.

When there are several SaF middleboxes in the path between S and R, it is necessary to assume that the SaF 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.

Some practical use cases when this trust model is likely to apply are given in Appendix A.

The security requirements for SRTP SaF hence are:

1. It SHALL be possible to provide e2e confidentiality and message authentication between S and R.

2. It SHALL be possible to provide hbh message authentication between S and M, between M and R, and between two middleboxes M, M'.

To provide a basis for enhanced privacy protection against other parties (e.g. traffic analysis), hbh confidentiality SHOULD also be provided.

As mentioned, hbh cryptographic processing is compatible with SRTP [RFC3711] and therefore also RTCP SHOULD be protected using SRTCP on hbh basis according to [RFC3711]. However, the SRTP SaF extension defined herein makes no provision to provide protection of RTCP also on e2e basis. Relevant considerations (rationale and caveats) relating to RTCP are provided in Section 4.6.

3.3. Requirements on e2e Key Management

The following applies to the generic Store-and-Forward problem. For specifics about SRTP SaF key management, see Sections 4.4.2 and 5.5. In the following we assume that the hbh and e2e key management are handled independently of each other.. As the receiver needs to map the received packets to a cryptographic context, the middlebox needs to store information that can later identify the context. This information can either be the entire context or an identifier for the context.

If the entire context is transferred via the middlebox, the context needs to be protected with a shared (sender-receiver) secret key or the receiver's public key. As no negotiation or message exchange influencing the key generation can be assumed to take place a half-roundtrip key management protocol needs to be used. The middle box may at most contribute some freshness nonce for the key derivation performed by the sender. The cryptographic context used by the sender must then be transferred to the receiver as is.

Credentials used to protect the cryptographic context against intercept by the middlebox may have been (pre-)shared by direct communication (sender-receiver) or obtained via a trusted third party

(PKI or some corresponding secret key infrastructure).

If an identifier for the context is used, the context itself can be exchanged before, concurrently with, or after the message has been stored. Context exchange before storage places no specific requirements on the key management; any existing key management protocol could be used as long as it is possible to bind the generated cryptographic context to the identifier. Context exchange concurrent with storage requires that sender and receiver are both online. This is reasonable for media distribution use cases, but not for answering machine use cases. Context exchange after message storage requires that the receiver has no influence on the session key, so key management protocols requiring negotiation (e.g. Diffie-Hellman) cannot be used. If sender and receiver are online at the same time, the context exchange can be done directly between them and otherwise it can be done via a trusted third party.

3.4. Problems with SRTP in SaF 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 recorded and later resent by an middlebox in general are independent; received SRTP-encrypted payloads cannot just be stored and later 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 SaF scenario, namely SSRC and receiver’s IP address and port.

The SRTP SaF extension defined in this document addresses these problems.

3.5. Design Rationale

As noted above, different SaF scenarios 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 SaF 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 SaF 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 SaF use case 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 SaF use cases, which, by
- simple configuration of the extended framework (Section 4.3) can be adopted to support the requirements of the specific SaF 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 SaF extension compared to having per-SaF-use-case extensions.

4. Usage of SaF Security within SRTP

4.1. The SaF Extension

The SaF extension consists of a new packet format (Section 4.3), an extended cryptographic context concept (Section 4.4), and new SRTP processing at sender/receiver (Section 4.5). Considering only the cryptographic processing, SaF middleboxes are compatible with [RFC3711], and the necessary additional processing is defined in Section 4.5.2. Senders/receivers need to support new cryptographic transforms (see Section 4.7).

4.2. Terminology

A SaF e2e session is defined as the set of SaF e2e protected data produced under a single e2e context (a security association between sender and the ultimate receiver, see Section 4.4.1 for the exact definition of e2e context). A SaF e2e session may comprise several so-called SaF sources, i.e. several distinct logical e2e media streams to be protected by the same e2e context.

A SaF hbh session is defined as the set of SaF hbh protected data produced under a single hbh context (a security association between two entities, see Section 4.4 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.

4.3. SRTP SaF Packet Format

Figure 2 illustrates the format of the SRTP packet when SaF is applied.

The packet format is composed of an "inner" e2e (sender-receiver) part embedded in an "outer" hbh (sender-middlebox or middlebox-receiver) part. Between these parts, a new CCI field (explained below) is introduced.

The e2e protected portion provides e2e encryption of the payload, RTP padding, and RTP pad count. The e2e protected portion also defines two new fields (PUV and SSS) for cryptographic synchronization, and an e2e MAC tag field.

The e2e MAC tag covers the e2e protected portion, except the e2e MAC tag itself. Whether authentication implies source origin authentication or only message integrity depends on the transform

used. Thus, e2e encryption is provided over the Payload, RTP padding, and RTP pad count fields, while authentication is provided for the PUV and SSS as well.

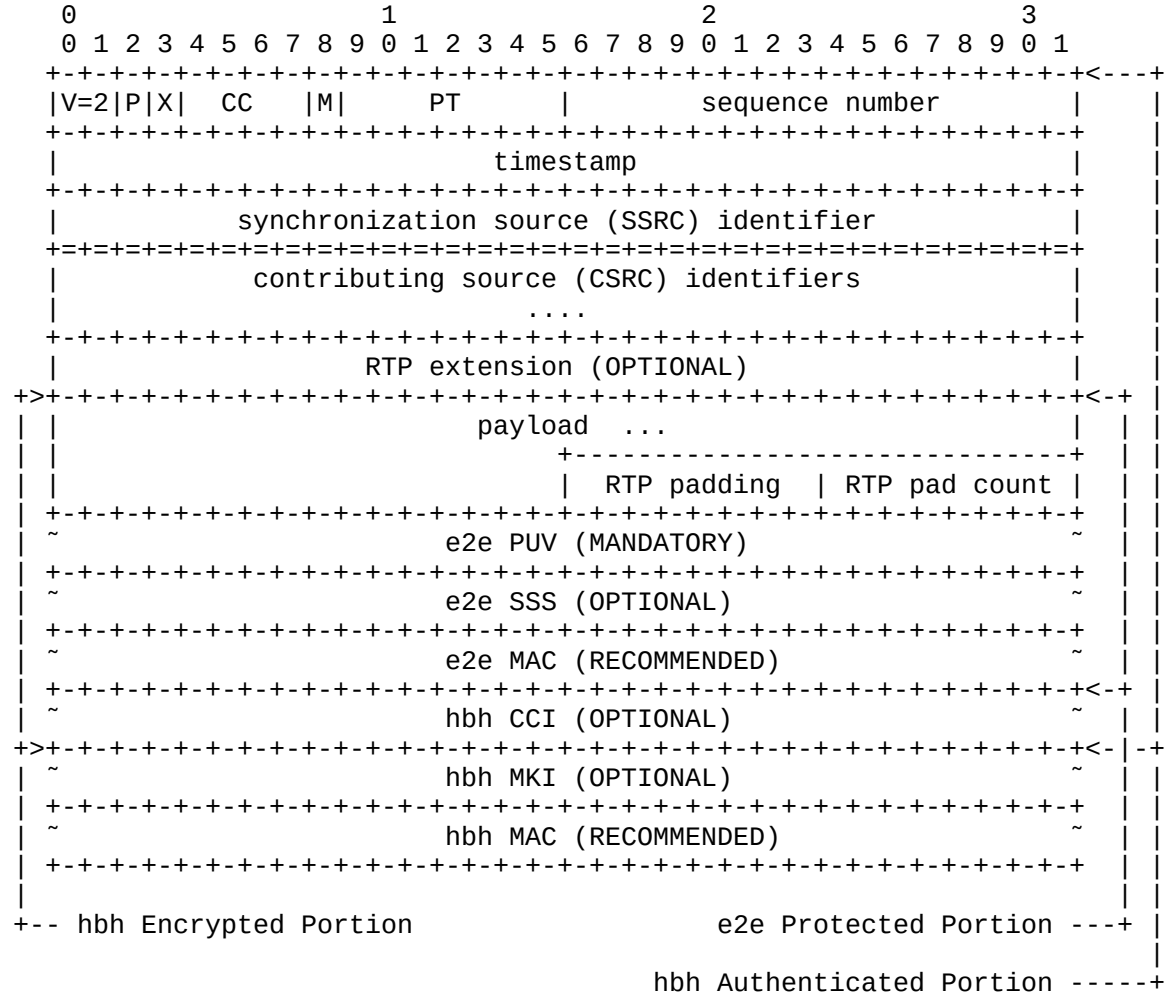

Figure 2: The format of the SRTP packet when SaF is applied.

Default e2e transforms, which provide both encryption and authentication, and which SHALL be supported are defined in Section 4.7.3.

The e2e protected portion is opaque from SaF middlebox point-of-view.

Thus, by treating the inner e2e protected portion and the Crypto

Context Identifier (CCI, see below) as the (hbh) "encrypted portion" of [RFC3711], the overall SRTP SaF 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 SaF Source is a value used by the SRTP SaF transform as an identifier for the SaF source within a SaF e2e session. Thus, SSS MUST be unique for all SaF sources within the SaF e2e session. Since there may be only one such SaF 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 4.7 and 5.2. 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 can be used if there are n SaF 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 5.2 and 5.4. For a given SaF e2e session and SaF 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 encryption transform.

- MAC (e2e): This field is used to carry payload authentication data e2e. It is transform dependent, of configurable length and is RECOMMENDED to be used. Observe that the e2e MAC SHALL cover the RTP payload, the PUV and SSS but SHALL NOT cover the RTP header, nor the CCI.

- CCI: Crypto Context Identifier: used to signal hbh, which e2e cryptographic context (keys and other parameters, see Section 4.4.1) to use. The field is OPTIONAL and of configurable length. The format is implementation specific, but the values 0, 1, 2, ..., n-1 can be used if there are n e2e contexts.

Parameters which are configurable have default values (see Section 4.7.4), and are otherwise negotiated during SaF 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 4.1).

Field	SRTP Counterpart	Typical Size (bytes)
PUV	SRTP Index	3
SSS	SSRC (IV formation)	0-1
MAC (e2e)	MAC (hbh)	4-10
CCI	SSRC (context identification)	0-1
Total		7-15

Table 4.1: Additional parameters

4.4. Extension of the SRTP Cryptographic Context

A SRTP SaF 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 SaF middlebox, between SaF middlebox and receiver, or, between two SaF middleboxes. The hbh context SHALL thus be identified by the <SSRC, destination network address, destination port number> triplet exactly as defined in [RFC3711].

2. One (or more) e2e contexts: this part of the context is defined below and SHALL be used for the e2e protection between sender and receiver.

The motivation for allowing more than one e2e context is to support scenarios where the SaF middlebox and receiver use a single (S)RTP session into which several e2e protected sessions are spliced, see Appendix A for use cases.

If the SRTP SaF context contains more than one e2e context, then each e2e context SHALL be associated with a unique CCI value. Since the length of the CCI field is variable, the length of the CCIs SHALL be determined by a length parameter, n_CCI.

4.4.1. Definition of e2e Context

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

- an identifier for the e2e encryption algorithm, i.e., the cipher and its mode of operation, see Section 4.7.3 for the default e2e encryption transform specification,

- an identifier for the e2e message authentication algorithm, see Section 4.7.3 for the default e2e message authentication 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. 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, n_a, n_s, and n_tag determining the length of the e2e session keys for encryption and message authentication, the e2e session salt, and the e2e authentication tag,
- non-negative integers n_PUV, and n_SSS determining the length of the PUV, and SSS fields.

There may also be need to include e2e transform dependent parameters, see Section 4.7.3 for the parameters associated with the default e2e transforms.

Observe that there is no replay protection data in the e2e context, see Section 4.5.3.1. 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 4.6.

Only end-points need to support e2e contexts, i.e. senders and receivers. SaF middleboxes need, however, to understand the usage of the e2e context identifiers (CCI) as discussed next.

If a SaF Middlebox needs to send information to the receiver that is not e2e protected and not associated with an e2e context beforehand, the SaF middlebox SHALL create an associated e2e context and follow the processing steps for SRTP SaF. A typical use-case is e.g. an answering machine sending supporting information in-band to the receiver, e.g. "You have three new messages", or, to the sender, e.g. "Please leave a message after the tone".

4.4.2. Identification of e2e Context

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

4.4.2.1. 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.

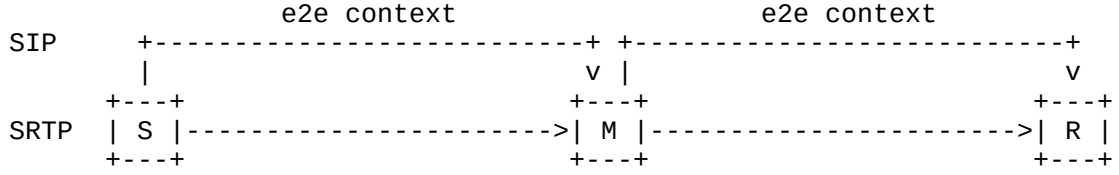

Figure 3: Transferring the entire e2e context via the middlebox

Alternatively, out-of-band context identification may use indirection and SHALL then be defined as follows. For each e2e context, a Content ID (CID) is defined. When used, the CID MUST uniquely determine the context between a sender and a receiver but the exact format of the CID is outside the scope of this specification. For example, a statistically unique (e.g. 256-bit) value may be used. The CID is communicated by out-of-band means:

- e2e (together with the e2e context), directly between sender and receiver or via a trusted third party.
- hbh, between sender and SaF middlebox, between two SaF middleboxes (as applicable), and between SaF middlebox and receiver.

How the CID communication is done (which protocol to use etc.) is outside the scope of this specification. Any SRTP key management protocol (e.g. DTLS-SRTP) can be used. If the e2e communication is done directly between sender and receiver, they must be simultaneously online, which is reasonable in many use cases (e.g. media distribution). The e2e communication MAY be done before or after the communication with M.

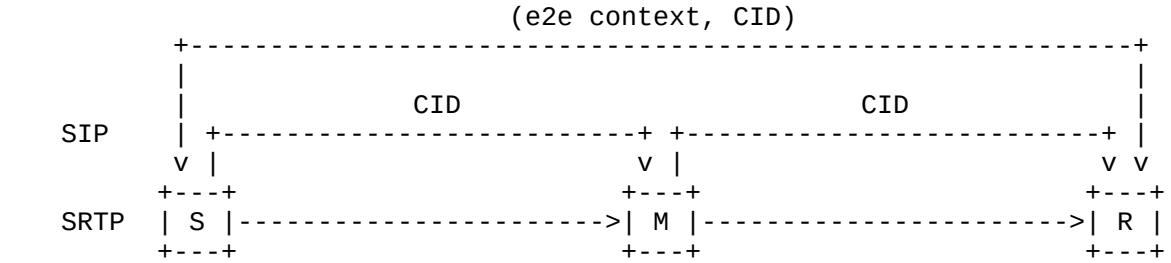

Figure 4: Indirection via CID

In the sequel, the term CID will often be used instead of "CID/e2e context".

4.4.2.2. 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 quadruplet context identifier:

<CCI, SSRC, destination network address, destination port number>

The e2e context (either the context itself or its CID) is here implicit from the quadruplet. If the SaF context contains more than one e2e context, the triplet context identifier [RFC3711] cannot uniquely identify the e2e context and the context identifier has therefore been extended with the CCI field in the SRTP SaF packet (see Figure 2). Just like in [RFC3711] the entire quadruplet MAY not be needed to uniquely identify the e2e context. Note that different quadruplets identify the e2e context on different "hops".

The CCI may thus be thought of as a short, in-band alias for the e2e context (possibly via additional indirection through CID) and is only used on hbh basis. If multiple pieces of content corresponding to multiple CIDs are transferred within the same SaF hbh session, the source (initial sender or SaF middlebox) SHALL ensure the use of distinct CCIs for all CIDs.

4.4.2.3. Mapping CCI to CID

The SaF middlebox SHALL, in conjunction to informing the next hop destination about the CID values, also inform if and how it has associated the CCIs to CIDs, e.g. as part of session setup signaling. For instance, the SaF middlebox MAY provide pairs of form:

(CID1, CCI1), (CID2, CCI2), ...

During transfer of e2e protected content associated with a certain CID, the source (initial sender or SaF middlebox) SHALL add the associated CCI to each packet being part of that content.

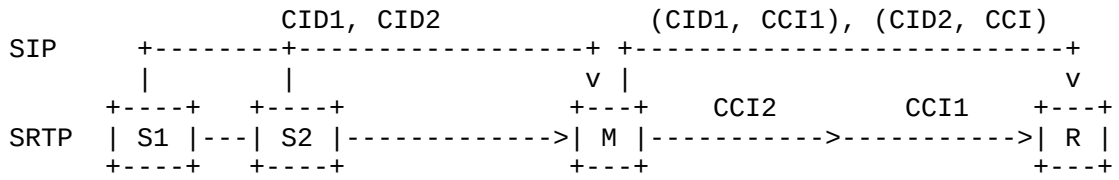

Figure 5: Switching between e2e contexts

Figure 5 illustrates CCI to CID mapping and e2e context switching. The senders S1 and S2 stores messages for the same receiver R on a SaF middlebox M. They specify different CIDs (CID1, CID2) in the signaling to M. When R later retrieves the content from M it is multiplexed inside the same hbh session. Different CCIs are used to identify the e2e contexts. Before the messages are sent, M informs R about the CIDs and their corresponding CCIs.

4.5. SRTP SaF Processing

In what follows, it is assumed that the sender and receiver 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.

4.5.1. Sender

The sender SHALL first, out-of-band, establish the necessary CIDs, CCIs, and hbh context parameters with the SaF 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 4.4. Next, and prior to performing step 2 of [RFC3711], the sender SHALL perform step S2-S6 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 4.7.2.

S3 The sender SHALL next apply the e2e encryption transform as described in Section 4.7.3.

S4 The sender SHALL next apply the e2e authentication transform as described in Section 4.7.3 to the result of S3 concatenated by the PUV and the SSS.

S5 The sender SHALL then form the e2e protected portion of the SRTP SaF packet by concatenating the result of S3, the PUV, the SSS, and the e2e MAC from S4.

S6 The sender SHALL then concatenate the result of S5 and the CCI.

The rest of the sender's processing conforms to [RFC3711], steps 2-8, by treating the result of S6 as the part to be encrypted ("encrypted portion" of [RFC3711]) and using the hbh context.

4.5.2. SaF Middlebox

SaF 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 SaF middlebox SHALL first, out-of-band, establish the necessary CIDs, CCIs, and hbh context parameters with the source or destination.

4.5.2.1. Acting as Receiver ("Store")

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

MR2 The SaF middlebox SHALL store sufficient information to later be able to map the correct content to the intended receiver, e.g. e2e context, the CID, 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 SaF middlebox SHALL store information sufficient to later reconstruct the e2e protected portion of the packets (corresponding to Figure 2) and to allow the receiver to uniquely identify the correct e2e context, e.g. by storing the CID or 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 SaF middlebox.

4.5.2.2. Acting as Sender ("Forward")

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

MS2 A payload SHALL be formed by concatenating the e2e protected portion and the CCI.

MS3 MS3 An RTP Header SHALL be formed with the P, M, and PT fields identical to when the payload was stored. The original timestamp MAY be used, but if several messages are spliced together, an offset needs to be added to that the hbh timestamps form a continuous sequence. The SEQ and SSRC SHOULD be defined strictly on hbh basis.

MS4 The SaF 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].

4.5.2.3. Multiple SaF Middleboxes

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

M1 SHALL act as a SaF middlebox sender (Section 4.5.2.2) treating M2 as a receiver. M2 SHALL act as a SaF middlebox receiver (Section 4.5.2.1) treating M1 as a sender.

4.5.3. Receiver

R1 The receiver SHALL first, out-of-band, establish the necessary CIDs, CCIs, and hbh context parameters with the SaF 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 and the CCI 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 4.4.2. (In case the CCI was NOT used or NOT encrypted by the hbh transform, the receiver MAY determine the e2e context already in step R1.)

R4 The receiver SHALL derive the e2e session encryption/ authentication key(s) as describe in Section 4.7.2 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 4.7.3. 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 5.4).

R6 The receiver removes PUV, SSS, e2e MAC, and CCI as appropriate.

4.5.3.1. 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 5.4 for security considerations.

4.6. Use of SRTCP with SRTP SaF

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 SaF.

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 SaF middlebox to reproduce RTCP reports accurately reflecting the ongoing SaF 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.

4.7. Cryptographic Transforms

We define a set of pre-defined SRTP SaF e2e transforms. Note that SaF 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 4.7.1, or, transforms that have been added later (see Section 4.7.5). When separate e2e transforms are used for encryption and authentication, the sender SHALL first apply the e2e encryption transform and then the e2e authentication transform.

4.7.1. Pre-Defined e2e Transforms

The pre-defined e2e encryption transform is AES-CM (AES Counter Mode) as specified in [RFC3711], Section 4.1.1, with the following modification. Instead of forming the initialization vector as defined in [RFC3711], the IV SHALL be formed as:

$$IV = (k\_s * 2^{16}) \text{ XOR } (SSS * 2^{64}) \text{ XOR } (PUV * 2^{16})$$

where k_s is the session salting key (derived from the e2e master key and salt, see Section 4.7.2) and where 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-CM 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-CM SHALL be 64 bits.

The key used SHALL be the session encryption key k_e (derived from the e2e master key and salt, see Section 4.7.2).

The pre-defined e2e authentication transform is HMAC-SHA1 as defined in [RFC3711], Section 4.2.1, with the difference that it SHALL be applied to the e2e protected portion, excluding the e2e MAC field itself. Note also that the e2e MAC SHALL NOT be applied to the CCI field. The resulting MAC tag SHALL be inserted in the e2e MAC field.

The key used SHALL be the session authentication key k_a (derived from the e2e master key and salt, see Section 4.7.2).

If a key management protocol designed to key the original SRTP specification [RFC3711] is used to set up the e2e part of SRTP SaF, the values that specify AES-CM and HMAC-SHA1 in [RFC3711] SHALL specify the transforms defined in Section 4.7.1.

4.7.2. 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.

4.7.3. 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 encryption transform SHALL be AES-CM as defined in Section 4.7.1. The default e2e authentication transform SHALL be HMAC-SHA1 as defined in Section 4.7.1. The default e2e pseudo-random function SHALL be AES-CM as defined in [RFC3711], Section 4.3.3.

4.7.4. SRTP SaF 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 parameters:

parameter	min	default
n_PUV	16	24
n_SSS	0	0
n_CCI	0	0

Table 4.2: Additional parameters

4.7.5. 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. An authentication transform MUST define how the e2e MAC is computed and MUST NOT include the CCI field in the authentication coverage.

When separate transforms are used for encryption and authentication, the sender SHALL first apply the e2e encryption transform and then the e2e authentication transform. When a combined (data encapsulation) transform is used, the order of processing is typically built in to the transform.

5. Security Considerations

5.1. 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 then one may think and the security considerations turn out to be essentially equivalent.

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

In this specification three new in-band parameters, PUV, CCI, and SSS, are specified. Note that CCI and SSS are used in exactly the same way the SSRC is used in [RFC3711]: context identification (CCI) and IV formation (SSS). Basically, one can think of the CCI as the e2e context identifier and the SSS as the e2e source identifier. The SSS is e2e protected. As the e2e authentication fails if the CCI is modified it does not need to be e2e protected (similar to MKI [RFC3711]).

5.2. 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 SaF middlebox. Another situation when key reuse may be beneficial is if sender and receiver use the SaF 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.

5.3. Authentication and Authorization

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

5.4. Replay Protection

Replay protection is provided on an hbh basis by use of a 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 SaF 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 SaF 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.

5.5. 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 5.2 for discussion on keystream reuse).

5.6. Privacy

In order for a SaF middlebox to deliver the correct media (produced with the correct e2e context) to the receiver, some SaF 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. Such techniques are to use random CIDs, and to assign CCIs independent on the order the messages were stored. Complete anonymity may be in conflict with the requirement that the SaF middlebox needs protection from flooding by garbage or other forms of unwanted traffic.

5.7. RTCP Considerations

As specified, RTCP is only protected on hbh basis. This is motivated by the assumption that a SaF 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.

5.8. Malicious middleboxes

SaF 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 SaF 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 network operator) to perform such attacks to its paying customers.

6. Acknowledgements

The authors would like to thank Daniel Catrein, Steffen Fries, Frank Hartung, Joerg Ott, and Magnus Westerlund for their support and valuable comments. We are also grateful to Eric Rescorla for his feedback when reviewing the -00 version of this draft.

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

[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.

Appendix A. Use Cases

In the use cases below, we map the entities to the trust model of Section 3.2 by indicating which entity that corresponds to S, M, R.

A.1. Streaming Pre-encrypted Media

A content creator (S) wants to distribute high value content to clients (R). The content provider distributes the media via a streaming server (M) which should not have access to cleartext media, typically because it is not trusted by the content creator.

A.2. Recording Encrypted Media at Home

High value encrypted media (e.g. IPTV, and radio) is broadcasted in a network. Only clients trusted by the content creator (S) have access to the encryption key. A user (R) is recording the media on a Hard Disk Drive (M), but does not yet have a license or have a license that does not allow cleartext copying. The media is therefore stored in protected format on the HDD.

A.3. Answering Machine

Operators commonly provide an answering machine service to their customers. In this case the communicating parties (S and R) may not wish to disclose the media to any other party, and hence want to apply encryption between each other. The answering machine (M) acts as a SaF middlebox, which has to store encrypted data and retransmit it to the callee.

In this use case it is also likely that several callers leave messages protected by different e2e keys. As discussed in the SRTP SaF specification, the receiver and SaF middlebox may agree to use a single hbh context into which the different e2e contexts are multiplexed using the CCI.

A.4. Media Rewind

Common to the use cases above is the possible desire to be able to rewind or jump forward in the media stream. For instance, a user may wish to listen once again to a message left in a voice mail without terminating and reinitiating the session with the SaF middlebox.

Appendix B. Test Vector

The parameters are chosen to be typical for a voice call. A frame size of 32 bytes is used by AMR 12.2 (Adaptive Multi-Rate) and with 20 ms speech frames, a PUV length of 24 bits equals 93.2 h. A 32-bit MAC offers good integrity protection for a voice call. The 16 bit SSS is not typical but is included to make the test vector more general.

```
Encryption algorithm:                    AES-CM
Authentication algorithm:                HMAC-SHA-1
Pseudo Random Function:                  AES-CM

n_e (encr session key length):           128
n_a (auth session key length):           160
n_s (session salt key length):           112
n_PUV (Packet Unique Value length):       24
n_SSS (SRTP SaF Source length):           16
n_tag (Authentication tag length):        32
```

The values below are in hexadecimal.

```
e2e Master key (128 bits)
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
```

```
   e2e Master salt (112 bits)
   40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d

   e2e Session encryption key (128 bits)
   12 ed 05 3a f7 8c 9a f2 96 5c 64 26 f4 d1 56 23

   e2e Session authentication key (160 bits)
   73 0c 3c ac 1d 75 27 36 91 97 d4 ab c2 b4 6b 46
   cd e0 19 83

   e2e Session salting key (112 bits)
   eb 31 d1 cb af 09 68 cd 14 f2 2b be 35 18

   Packet Unique Value (24 bits)
   80 81 82

   SRTP SaF Source (16 bits)
   c0 c1

   Payload (256 bits)
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

   e2e Protected Portion (328 bits)
   82 37 69 bd f8 9c f3 61 57 e4 3d 74 b7 e6 07 4b
   05 80 52 ec 7d 68 72 63 b2 e1 10 ae b9 7b 7c a0
   80 81 82 c0 c1 bd ab 1e f6
```

Authors' Addresses

Rolf Blom
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 31 707
Email: rolf.j.blom@ericsson.com

Yi Cheng
Ericsson AB
SE-164 80 Stockholm
Sweden

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

Fredrik Lindholm
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 31 705
Email: fredrik.lindholm@ericsson.com

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

Karl Norrman
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 44 502
Email: karl.norrman@ericsson.com

Network Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: August 18, 2011

M. Petit-Huguenin
Stonyfish, Inc.

February 14, 2011

Support for multiple clock rates in an RTP session
draft-petithuguenin-avtext-multiple-clock-rates-00

Abstract

This document clarifies the RTP specification when different clock rates are used in an RTP session. It also provides guidance on how to interoperate with legacy RTP implementations that use multiple clock rates.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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 August 18, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The clock rate is a parameter of the payload format. It is often defined as been the same as the sampling rate but it is not always the case (see e.g. the G722 and MPA audio codecs in [RFC3551]).

An RTP sender can switch between different payloads during the lifetime of an RTP session and because clock rates are defined by payload types, it is possible that the clock rate also varies during an RTP session. RTP [RFC3550] lists using multiple clock rates as one of the reasons to not use different payloads on the same SSRC but unfortunately this advice was not always followed and some RTP implementations change the payload in the same SSRC even if the different payloads use different clock rates.

This creates three problems:

- The method used to calculate the RTP timestamp field in an RTP packet is underspecified.
- When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the RTP timestamp field in an RTCP SR packet.
- When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the interarrival jitter field in an RTCP RR packet.

Table 1 contains a non-exhaustive list of fields in RTCP packets that uses a clock rate as unit:

Field name	RTCP packet type	Reference
RTP timestamp	SR	[RFC3550]
Interarrival jitter	RR	[RFC3550]
min_jitter	XR Summary Block	[RFC3611]
max_jitter	XR Summary Block	[RFC3611]
mean_jitter	XR Summary Block	[RFC3611]
dev_jitter	XR Summary Block	[RFC3611]
Interarrival jitter	IJ	[RFC5450]
RTP timestamp	SMPTETC	[RFC5484]
Jitter	RSI Jitter Block	[RFC5760]
Median jitter	RSI Stats Block	[RFC5760]

Table 1

This document changes the RTP specification by recommending to use a different SSRC for each clock rate in most of the cases.

2. Legacy RTP

The following sections describe the various ways legacy RTP implementations behave when multiple clock rates are used. Legacy RTP refers to RFC 3550 without the modifications introduced by this document.

2.1. Different SSRC

One way of managing multiple clock rates is to use a different SSRC for each different clock rate, as in this case there is no ambiguity on the clock rate used by fields in the RTCP packets. This method also seems to be the original intent of RTP as can be deduced from points 2 and 3 of section 5.2 of RFC 3550.

On the other hand changing the SSRC can be a problem for some implementations designed to work only with unicast IP addresses, where having multiple SSRCs is considered a corner case. Lip synchronization can also be a problem in the interval between the beginning of the new stream and the first RTCP SR packet. This is not different than what happen at the beginning of the RTP session but it can be more annoying for the end-user.

2.2. Same SSRC

The simplest way of managing multiple clock rates is to use the same SSRC for all the payload types regardless of the clock rates.

Unfortunately there is no clear definition on how the RTP timestamp should be calculated in this case. The following subsections present two variants.

2.2.1. Monotonic timestamps

The most common method of calculating the RTP timestamp ensures that the value increases monotonically. The formula used by this method is as follow:

timestamp = previous_timestamp + (current_capture_time - previous_capture_time) * current_clock_rate

The problem with this method is that the jitter calculation on the receiving side gives invalid result during the transition between two clock rates, as shown in Table 2. The capture and arrival time are in seconds, starting at the beginning of the capture of the first packet; clock rate is in Hz; the RTP timestamp does not include the random offset; the transit, jitter, and average jitter use the clock rate as unit.

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	800	0.18	2080	480	30
0.1	16000	1120	0.2	2080	0	28
0.12	16000	1440	0.22	2080	0	26
0.14	8000	1600	0.24	320	720	70
0.16	8000	1760	0.26	320	0	65

Table 2

Calculating the correct transit time on the receiving side can be done by using the following formulas:

(1) current_time_capture = current_timestamp - previous_timestamp) / current_clock_rate + previous_time_capture
(2) transit = current_clock_rate * (time_arrival - current_time_capture)
(3) previous_time_capture = current_time_capture

The main problem with this method, in addition to the fact that the jitter calculation described in RFC 3550 cannot be used, is that is it dependent on the previous RTP packets, packets that can be reordered or lost in the network. But it seems that this is what most implementations are using.

2.2.2. Non-monotonic timestamps

An alternate way of generating the RTP timestamps is to use the following formula:

timestamp = capture_time * clock_rate

With this formula, the jitter calculation is correct but the RTP timestamp values are no longer increasing monotonically as shown in Table 3. RFC 3550 states that "[t]he sampling instant MUST be derived from a clock that increments monotonically[...]" but nowhere says that the RTP timestamp must increment monotonically.

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	1280	0.18	1600	0	0
0.1	16000	1600	0.2	1600	0	0
0.12	16000	1920	0.22	1600	0	0
0.14	16000	2240	0.24	1600	0	0
0.16	16000	2560	0.26	1600	0	0
0.14	8000	1120	0.24	800	0	0
0.16	8000	1280	0.26	800	0	0

Table 3

The advantage with this method is that it works with the jitter calculation described in RFC 3550, a long as the correct clock rates are used.

3. 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 [RFC2119].

Clock rate: The multiplier used to convert from a wallclock value in seconds to an equivalent RTP timestamp value (without the fixed random offset). Note that RFC 3550 uses various terms like "clock frequency", "media clock rate", "timestamp unit", "timestamp frequency", and "RTP timestamp clock rate" as synonymous to clock rate.

RTP Sender: A logical network element that sends RTP packets, sends RTCP SR packets, and receives RTCP RR packets.

RTP Receiver: A logical network element that receives RTP packets, receives RTCP SR packets, and sends RTCP RR packets.

4. RTP Sender

An RTP Sender with RTCP turned off (i.e. by setting the RS and RR bandwidth modifiers defined in [RFC3556] to 0) SHOULD use a different SSRC for each different clock rate but MAY use different clock rates on the same SSRC as long as the RTP timestamp is calculated as

described in Section 2.2.2.

An RTP Sender with RTCP turned on MUST use a different SSRC for each different clock rate. An RTCP BYE MUST be sent and a new SSRC MUST be used if the clock rate switches back to a value already seen in the RTP stream.

To accelerate lip synchronization, the next compound RTCP packet sent by the RTP sender MUST contain multiple SR packets, the first one containing the mapping for the current clock rate and the next SR packets containing the mapping for the other clock rates seen during the last period.

The RTP extension defined in [RFC6051] MAY be used to accelerate the synchronization.

5. RTP Receiver

An RTP Receiver MUST be able to handle clock rate changes either on the same SSRC (Section 2.1) or on different SSRC (Section 2.2.2).

An RTP Receiver MUST be able to handle a compound RTCP packet with multiple SR packets.

For interoperability with legacy RTP implementations, an RTP receiver MAY use the information in two consecutive SR packets to calculate the clock rate used, i.e. if Ni is the NTP timestamp for the SR packet i, Ri the RTP timestamp for the SR packet i and Nj and Rj the NTP timestamp and RTP timestamp for the previous SR packet j, then the clock rate can be guessed as the closest to (Ri - Rj) / (Ni - Nj).

6. Interoperability Analysis

The next subsections analyze the various combinations between legacy RTP implementations and RTP implementations that follow this document specifications.

6.1. Legacy RTP Sender using different SSRC sending to new RTP Receiver

Because a specific clock rate is associated to a specific SSRC, there is no ambiguity in the RTP timestamp received in the RTP packet or SR packet or in the jitter sent in the RR packet.

6.2. Legacy RTP Sender using same SSRC with monotonic timestamps sending to new RTP Receiver

The new RTP Receiver will not be able to rebuild the correct RTP timestamp so the jitter will be incorrect. Note that this is not different than if a legacy RTP Receiver is used.

6.3. Legacy RTP Sender using same SSRC with non-monotonic timestamps sending to new RTP Receiver

TBD

6.4. New RTP Sender using different SSRC sending to legacy RTP Receiver

Because a specific clock rate is associated to a specific SSRC, there is no ambiguity in the RTP timestamp received in the RTP packet or SR packet or in the jitter sent in the RR packet. Some legacy RTP implementations may have problems when receiving multiple SR packets.

6.5. New RTP Sender using different SSRC sending to new RTP Receiver

Because a specific clock rate is associated to a specific SSRC, there is no ambiguity in the RTP timestamp received in the RTP packet or SR packet or in the jitter sent in the RR packet.

6.6. New RTP Sender using same SSRC with non-monotonic timestamps to legacy RTP Receiver

Because this combination is used only when no RTCP packets are exchanged, there is no problem interpreting the RTCP field units. Some legacy RTP implementations may have problems if the jitter clock rates are not correctly managed.

6.7. New RTP Sender using same SSRC with non-monotonic timestamps to new RTP Receiver

Because this combination is used only when no RTCP packets are exchanged, there is no problem interpreting the RTCP field units.

7. Security Considerations

TBD

8. IANA Considerations

No IANA considerations.

9. Acknowledgements

Thanks to Colin Perkins and Ali C. Begen for their comments, suggestions and questions that helped to improve this document.

Thanks to Robert Sparks and the attendants of SIPit 26 for the survey on multiple clock rates interoperability.

This document was written with the xml2rfc tool described in [RFC2629].

10. References

10.1. 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.

10.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast

Sessions with Unicast Feedback", RFC 5760, February 2010.

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

[uRTR] Wenger, S. and C. Perkins, "RTP Timestamp Frequency for Variable Rate Audio Codecs", draft-ietf-avt-variable-rate-audio-00 (work in progress), October 2004.

Appendix A. Using a fixed clock rate

An alternate way of fixing the multiple clock rates issue was proposed in [uRTR]. This document proposed to define a unified clock rate, but the proposal was rejected at IETF 61.

Appendix B. Release notes

This section must be removed before publication as an RFC.

B.1. Modifications between draft-petithuguenin-avtext-multiple-clock-rates-00 and draft-petithuguenin-avt-multiple-clock-rates-03

- Initial release for avtext WG.

B.2. Modifications between draft-petithuguenin-avt-multiple-clock-rates-03 and draft-petithuguenin-avt-multiple-clock-rates-02

- Updated RFC reference.

B.3. Modifications between draft-petithuguenin-avt-multiple-clock-rates-02 and draft-petithuguenin-avt-multiple-clock-rates-01

- Having multiple SRs in a compound RTCP packet is OK.
- If RTCP is used, must send a BYE and not reuse the SSRC.
- Removed resolved notes.
- Acknowledged SIPit 26 survey.
- Fixed some nits.

B.4. Modifications between draft-petithuguenin-avt-multiple-clock-rates-01 and draft-petithuguenin-avt-multiple-clock-rates-00

- Complete rewrite as a Standard Track I-D modifying RFC 3550.

Author's Address

Marc Petit-Huguenin
Stonyfish, Inc.

Email: petithug@acm.org

AVTEXT Working Group
Internet-Draft
Intended status: Informational
Expires: August 30, 2011

J. Xia
Huawei

February 26, 2011

Content Splicing for RTP Sessions
draft-xia-avtext-splicing-for-rtp-00

Abstract

This memo outlines RTP splicing. Splicing is a process that replaces the content of the main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to receiver for a period of time. This memo discusses the requirements of splicing process on RTP layer, thus provides guideline for how a RTP Translator works to satisfy these requirements.

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 August 30, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction

This document outlines how splicing can be used for RTP sessions. Splicing is a process that replaces the content of the main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to receiver for a period of time. The substitutive content can be provided for example via another RTP stream or local media file storage.

One representative use case for splicing is targeted advertisements insertion, which allows operators to replace a national advertising slot with its own regional advertising content prior to providing to receiver. So far [SCTE30] and [SCTE35] have standardized MPEG2-TS splicing running over cable, but to date there is no guidelines for how to use the RTP Translator functionality defined in [RFC3550] to implement an RTP Splicer.

In this document, we describe the basic requirements of RTP splicing, then analyze how an RTP Translator can be used to implement RTP splicing to satisfy these requirements from the aspect of feasibility, implementation complexity and backward compatibility.

2. Terminology

The keywords "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].

Current RTP Stream

The RTP stream that the RTP receiver is currently receiving. The content of current RTP stream can be either main content or substitutive content.

Main RTP Stream

The RTP stream that the Splicer is receiving. The content of main RTP stream can be replaced by substitutive content for a period of time.

Main Content

The multimedia content that are conveyed in main RTP stream. main content will be replaced by the substitutive content during splicing period.

Substitutive Content

The multimedia content that replaces the content of main RTP stream during splicing period. The substitutive content can for example be contained in an RTP stream from a media source or fetched from local media file storage.

Insertion RTP Stream

A RTP stream that may provide substitutive content. If the substitutive content is provided from insertion RTP stream, the insertion RTP Stream must be terminated on Splicer.

Splicer

An intermediary node that inserts substitutive content into main RTP stream. Splicer sends substitutive content to RTP receiver as the payload of the current RTP stream.

3. RTP Splicing Discussion and Requirements

In this document, we assume an intermediary network element, which is referred to as Splicer, to play the key role to handle RTP splicing. When RTP splicing begins, Splicer replaces the main content in the main RTP stream with substitutive content and conveys the substituted RTP stream to RTP receiver for a period of time, then resumes the main content when RTP splicing finishes. The RTP splicing may happen more than once if substitutive content will be inserted in multiple time segments during the lifetime of the main RTP session.

To realize a seamless splicing on RTP receiver, Splicer must not cause any perceptible media clipping at the splicing joint. Therefore, main media source must reserve specific time slot in the main RTP stream for splicing and notify Splicer this specific time slot information. The substitutive content time length SHOULD be the same as the reserved time slot length. The details about how the specific time slot information is conveyed to Splicer are outside the scope of this memo.

Besides the media clipping avoidance, there are also a set of concrete requirements that must be satisfied on Splicer:

REQ-1:

Splicer MUST operate in either unicast or multicast session environment.

REQ-2:

Splicer MUST guarantee content substitution process is invisible to RTP receiver to prevent the RTP receiver from easily identifying the substitutive content.

The RTP packets whose payloads are replaced by substitutive content are required to keep their RTP header information to be consistent with those of main RTP packets whose payloads are unaltered:

The value of SSRC field in RTP header MUST be same as the value of corresponding field in main content RTP header, while the value of payload type field SHOULD be same as the value of corresponding field in main content RTP header. Note that in some cases, it may be necessary to transcode the substitutive content to ensure the payload type is the same, and that this may be prohibitively expensive, so it might be acceptable to leave the payload untranscoded.

The value of sequence number field in RTP header MUST be contiguous regardless of whether the substitutive content is inserted or not. It should be noted that the number of packets in the substitutive sequence number range may be different from the number of packets in the overridden main sequence number range due to the different characteristics or entropy coding.

REQ-3:

Splicer SHOULD ensure that the main media source can learn the real performance of the path between media source and downstream receiver for adaptation purpose.

REQ-4:

Splicer MUST be backward compatible with basic characteristics of [RFC3550], e.g., SSRC identifier collision resolution and loop detection.

REQ-5:

Splicer MUST have the capability to correctly handle any packet loss events regardless of whether the lost content is main content or substitutive content.

REQ-6:

If substitutive content comes from an insertion RTP stream, Splicer MUST terminate this stream, in such case, Splicer should generate RTCP reports upstream towards the insertion media source using its own SSRC.

4. Using RTP Translator for RTP Splicing

An RTP Translator that acts as media transcoder can replace the payloads of the main RTP packets with the substitutive content, and can assign new sequence numbers to the substituted packets with main RTP packets SSRC identifier intact. Note that the new sequence numbers of substituted RTP packets must seamlessly follow the sequence numbers of the previous main RTP packets. When splicing ends, Translator must switch back to the main RTP stream and output it to RTP receiver until next splicing occurs.

With respect to RTCP flow, Translator acting as a media transcoder will need to interpose itself into the RTCP flow as specified in section 7.2 of [RFC3550]. However, the substitutive content might have different characteristics compared to the main content it replaces. As a result, the translated RTCP Receiver Reports received by the main media source might be somewhat misleading for adaptation purposes, since they relate to other content that the main media source cannot control during the splicing period. Fortunately Translator has the capability to act as quality monitor and generate RTCP reports upstream towards main media source with its own SSRC thus reflecting the real characteristics of the main RTP stream and the reception quality on the Translator. These RTCP reports, as well as the translated RTCP reports sent from the downstream receiver, can provide main media source the general performance of the different segments of the path between main media source and RTP receiver. If the substitutive content is fetched from the insertion RTP stream, Translator acting as RTP receiver should generate RTCP receiver reports upstream towards the insertion media source to reflect the reception quality of the insertion RTP stream on the Translator.

When RTP receiver detects any packet loss during splicing, it may generate RTCP NACK message for packet loss recovery [RFC4585]. If Translator receives any RTCP NACK message from RTP receiver, Translator first needs to determine the sequence number range of lost packets requested by RTP receiver. Since parts of lost packets may contain substitutive content which does not relate to the main RTP stream, the translated RTCP NACK message might be somewhat misleading for packet loss recovery. Thus, when Translator is intercepting RTCP NACK packets, it should only pass those RTCP NACK packets that relate

to the relevant content upstream.

5. Processing Splicing on RTP Translator

Once Translator has learnt when to process splicing from main RTP source, it must get ready for the coming splicing in advance, e.g., fetches the substitutive content either from local media file storage or via insertion RTP stream. If the substitutive content comes from the insertion RTP stream, Translator must act as a RTP receiver and terminate this insertion RTP stream.

First the Translator should guarantee the media encoding of substitutive content to be same as the media encoding of main content. If the substitutive content uses other media encoding, Translator should perform media transcoding procedure for substitutive content.

When splicing begins, Translator replaces the main content with the substitutive content as if the substitutive content were sent from the main media source.

When splicing ends, Translator must switch back to main RTP stream, but since the number of packets in the substitutive sequence number may be different from the number of packets in the overridden main sequence number range, Translator still needs to modify the sequence numbers of subsequent main RTP packets prior to outputting them to RTP receiver even if no media transcoding occurs on main RTP stream. For subsequent current RTP packets, its start sequence number should be determined as being one after the end sequence number of previous substitutive RTP packets.

This is perhaps best explained by a pseudo code example:

```
when (splicing begin) {

the sequence number of the ith substitutive packet = the sequence
number of last main packet + i, until the splicing end;

}

when (splicing end) {

the sequence number of the following kth current packet = the
sequence number of last substitutive packet + k, until the next
splicing begin;

}
```

With regard to RTCP packets, Translator needs to rewrite RTCP Report. The "sender's byte count" field and the "sender's packet count" field in RTCP Sender Report should be changed. Translator then determines the proportion, P, of the number of packets Translator receives from main media source (numRev) to the number of packets Translator sends to RTP receiver (numSend) during a regular RTCP Reporting interval such that P = numRev / numSend. It should be noted that the value of P may be not fixed specially at the splicing joint, where the RTCP Reports may reflects the characteristics of the combination of main RTP packets and adjacent substitutive RTP packets. Once the proportion, P, is counted, the values of packet loss fields and the "extended last sequence number" field in RTCP Sender Report are scaled by dividing each of them by P, and reverse rewriting is made to the values of the corresponding fields in RTCP Receiver Report, i.e., multiplying each them by P. Meanwhile, Translator must generate RTCP Receiver Report upstream towards main media source using its own SSRC, reflecting the reception quality of the main RTP stream on the Translator. If the substitutive content comes from the insertion RTP stream, Translator acting as receiver must generate RTCP Receiver Report upstream towards insertion media source using its own SSRC.

If Translator receives any RTCP NACK message from RTP receiver for packet loss recovery, it first determines if the lost packets contain substitutive content. It is somewhat more complex that the lost packets requested in a single RTCP NACK message contain not only main content but also substitutive content. To address this, Translator must rewrite the translated RTCP NACK message into two separate RTCP NACK messages: one only requests the lost packets that contain main content, and is forwarded using the SSRC of that RTP receiver; and another only requests the lost packets that contain substitutive content, and is sent using the SSRC of the Translator if the substitutive content is delivered via insertion RTP stream, or fetching the lost substitutive content directly from corresponding local media file storage.

6. Implementation considerations

When Translator is used to handle RTP splicing, RTP receiver does not need any RTP/RTCP extension for splicing. As a trade-off, additional overhead could be induced on Translator compared to just translating the main RTP stream, since Translator must coordinate the switch between main content and substitutive content, and generate its own RTCP reports. Even if no new substitutive content will be inserted, the Translator still needs to modify the main RTP packets, and to recalculate the UDP/IP checksum until the main RTP session ends. If Translator serves multiple main RTP streams simultaneously, this may lead to large overhead on the Translator.

7. Security Considerations

The security considerations of the RTP specification [RFC3550], the Extended RTP profile for RTCP-Based Feedback [RFC4585], and the Secure Real-time Transport Protocol [RFC3711] apply. Translator must be trusted by main media source and insertion media source, and must be included in the security context.

8. IANA Considerations

No IANA actions are required.

9. Acknowledgments

The following individuals have reviewed the earlier versions of this specification and provided very valuable comments: Colin Perkins, Magnus Westerlund, Roni Even, Tom Van Caenegem, Joerg Ott, David R Oran, Cullen Jennings, Ali C Begen, and Ning Zong.

10. Change Log

10.1. draft-xia-avtext-splicing-for-rtp-00

The following are the major changes compared to previous version 00:

- Change primary RTP stream to main RTP stream, add current RTP stream as the streaming received by RTP receiver.
- Eliminate the ambiguity of inserted content with substitutive content which replaces the main content rather than pause it.
- Clarify the signaling requirements.
- Delete the description on Mixer and MCU in section 4, mainly focus on the direction whether a Translator can act as a Splicer.
- Add section 5 to describe the exact guidance on how an RTP Translator is used to handle splicing.
- Modify the security considerations section and add acknowledges section.

11. Normative References

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

[RFC2250] Hoffman, D., Fernando, G., Goyal, V., and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video", RFC 2250, January 1998.

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, 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.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[I-D.ietf-avt-ecn-for-rtp] Westerlund, M., "Explicit Congestion Notification (ECN) for RTP over UDP", draft-ietf-avt-ecn-for-rtp-03 (work in progress), October 2010.

[SCTE30] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Splicing API", 2001.

[SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2004.

[H.323] ITU-T Recommendation H.323, "Packet-based multimedia communications systems", June 2006.

Author's Address

Jinwei Xia
Huawei
Hui Hong Mansion
Nanjing, Baixia District 210001
China

Phone: +86-025-86622310
Email: xiajinwei@huawei.com