AVTCORE Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: September 07, 2015

C. S. Perkins
University of Glasgow
V. Singh
Aalto University

March 06, 2015

Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions

draft-ietf-avtcore-rtp-circuit-breakers-09

Abstract

The Real-time Transport Protocol (RTP) is widely used in telephony, video conferencing, and telepresence applications. Such applications are often run on best-effort UDP/IP networks. If congestion control is not implemented in the applications, then network congestion will deteriorate the user's multimedia experience. This document does not propose a congestion control algorithm; instead, it defines a minimal set of RTP "circuit-breakers". Circuit-breakers are conditions under which an RTP sender needs to stop transmitting media data in order to protect the network from excessive congestion. It is expected that, in the absence of severe congestion, all RTP applications running on best-effort IP networks will be able to run without triggering these circuit breakers. Any future RTP congestion control specification will be expected to operate within the constraints defined by these circuit breakers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is widely used in voice-over-IP, video teleconferencing, and telepresence systems. Many of these systems run over best-effort UDP/IP networks, and can suffer from packet loss and increased latency if network congestion occurs. Designing effective RTP congestion control algorithms, to adapt the transmission of RTP-based media to match the available network capacity, while also maintaining the user experience, is a difficult but important problem. Many such congestion control and media adaptation algorithms have been proposed, but to date there is no consensus on the correct approach, or even that a single standard algorithm is desirable.

This memo does not attempt to propose a new RTP congestion control algorithm. Rather, it proposes a minimal set of "circuit breakers"; conditions under which there is general agreement that an RTP flow is causing serious congestion, and ought to cease transmission. It is expected that future standards-track congestion control algorithms for RTP will operate within the envelope defined by this memo.

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]. This interpretation of these key words applies only when written in ALL CAPS. Mixed- or lower-case uses of these key words are not to be interpreted as carrying special significance in this memo.

3. Background

We consider congestion control for unicast RTP traffic flows. This is the problem of adapting the transmission of an audio/visual data flow, encapsulated within an RTP transport session, from one sender to one receiver, so that it matches the available network bandwidth. Such adaptation needs to be done in a way that limits the disruption to the user experience caused by both packet loss and excessive rate changes. Congestion control for multicast flows is outside the scope of this memo. Multicast traffic needs different solutions, since the available bandwidth estimator for a group of receivers will differ from that for a single receiver, and because multicast congestion control has to consider issues of fairness across groups of receivers that do not apply to unicast flows.

Congestion control for unicast RTP traffic can be implemented in one of two places in the protocol stack. One approach is to run the RTP traffic over a congestion controlled transport protocol, for example over TCP, and to adapt the media encoding to match the dictates of the transport-layer congestion control algorithm. This is safe for the network, but can be suboptimal for the media quality unless the transport protocol is designed to support real-time media flows. We do not consider this class of applications further in this memo, as their network safety is guaranteed by the underlying transport.

Alternatively, RTP flows can be run over a non-congestion controlled transport protocol, for example UDP, performing rate adaptation at the application layer based on RTP Control Protocol (RTCP) feedback. With a well-designed, network-aware, application, this allows highly effective media quality adaptation, but there is potential to disrupt the network's operation if the application does not adapt its sending rate in a timely and effective manner. We consider this class of applications in this memo.

Congestion control relies on monitoring the delivery of a media flow, and responding to adapt the transmission of that flow when there are signs that the network path is congested. Network congestion can be detected in one of three ways: 1) a receiver can infer the onset of congestion by observing an increase in one-way delay caused by queue build-up within the network; 2) if Explicit Congestion Notification (ECN) [RFC3168] is supported, the network can signal the presence of congestion by marking packets using ECN Congestion Experienced (CE) marks; or 3) in the extreme case, congestion will cause packet loss that can be detected by observing a gap in the received RTP sequence numbers. Once the onset of congestion is observed, the receiver has to send feedback to the sender to indicate that the transmission rate needs to be reduced. How the sender reduces the transmission rate is highly dependent on the media codec being used, and is outside the scope of this memo.

There are several ways in which a receiver can send feedback to a media sender within the RTP framework:

o The base RTP specification [RFC3550] defines RTCP Reception Report (RR) packets to convey reception quality feedback information, and Sender Report (SR) packets to convey information about the media transmission. RTCP SR packets contain data that can be used to reconstruct media timing at a receiver, along with a count of the total number of octets and packets sent. RTCP RR packets report on the fraction of packets lost in the last reporting interval, the cumulative number of packets lost, the highest sequence number received, and the inter-arrival jitter. The RTCP RR packets also contain timing information that allows the sender to estimate the network round trip time (RTT) to the receivers. RTCP reports are sent periodically, with the reporting interval being determined by the number of SSRCs used in the session and a configured session bandwidth estimate (the number of SSRCs used is usually two in a unicast session, one for each participant, but can be greater if the participants send multiple media streams). The interval between reports sent from each receiver tends to be on the order of a few seconds on average, although it varies with the session bandwidth, and sub-second reporting intervals are possible in high bandwidth sessions, and it is randomised to avoid synchronisation of reports from multiple receivers. RTCP RR packets allow a receiver to report ongoing network congestion to the sender. However, if a receiver detects the onset of congestion part way through a reporting interval, the base RTP specification contains no provision for sending the RTCP RR packet early, and the receiver has to wait until the next scheduled reporting interval.

- The RTCP Extended Reports (XR) [RFC3611] allow reporting of more complex and sophisticated reception quality metrics, but do not change the RTCP timing rules. RTCP extended reports of potential interest for congestion control purposes are the extended packet loss, discard, and burst metrics [RFC3611], [RFC7002], [RFC7097], [RFC7003], [RFC6958]; and the extended delay metrics [RFC6843], [RFC6798]. Other RTCP Extended Reports that could be helpful for congestion control purposes might be developed in future.

- Rapid feedback about the occurrence of congestion events can be achieved using the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] (or its secure variant, RTP/SAVPF [RFC5124]) in place of the RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the RTCP transmission rate keeps within its bandwidth allocation. It also defines transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. RTP Codec Control Messages [RFC5104] extend the RTP/AVPF profile with additional feedback messages that can be used to influence that way in which rate adaptation occurs, but do not further change the dynamics of how rapidly feedback can be sent. Use of the RTP/AVPF profile is dependent on signalling.

- Finally, Explicit Congestion Notification (ECN) for RTP over UDP [RFC6679] can be used to provide feedback on the number of packets that received an ECN Congestion Experienced (CE) mark. This RTCP extension builds on the RTP/AVPF profile to allow rapid congestion feedback when ECN is supported.

In addition to these mechanisms for providing feedback, the sender can include an RTP header extension in each packet to record packet transmission times. There are two methods: [RFC5450] represents the transmission time in terms of a time-offset from the RTP timestamp of the packet, while [RFC6051] includes an explicit NTP-format sending timestamp (potentially more accurate, but a higher header overhead). Accurate sending timestamps can be helpful for estimating queuing delays, to get an early indication of the onset of congestion.

Taken together, these various mechanisms allow receivers to provide feedback on the senders when congestion events occur, with varying degrees of timeliness and accuracy. The key distinction is between systems that use only the basic RTCP mechanisms, without RTP/AVPF rapid feedback, and those that use the RTP/AVPF extensions to respond to congestion more rapidly.

4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile

The feedback mechanisms defined in [RFC3550] and available under the RTP/AVP profile [RFC3551] are the minimum that can be assumed for a baseline circuit breaker mechanism that is suitable for all unicast applications of RTP. Accordingly, for an RTP circuit breaker to be useful, it needs to be able to detect that an RTP flow is causing excessive congestion using only basic RTCP features, without needing RTCP XR feedback or the RTP/AVPF profile for rapid RTCP reports.

RTCP is a fundamental part of the RTP protocol, and the mechanisms described here rely on the implementation of RTCP. Implementations that claim to support RTP, but that do not implement RTCP, cannot use the circuit breaker mechanisms described in this memo. Such implementations SHOULD NOT be used on networks that might be subject to congestion unless equivalent mechanisms are defined using some non-RTCP feedback channel to report congestion and signal circuit breaker conditions.

Three potential congestion signals are available from the basic RTCP SR/RR packets and are reported for each synchronisation source (SSRC) in the RTP session:

1. The sender can estimate the network round-trip time once per RTCP reporting interval, based on the contents and timing of RTCP SR and RR packets.

2. Receivers report a jitter estimate (the statistical variance of the RTP data packet inter-arrival time) calculated over the RTCP reporting interval. Due to the nature of the jitter calculation ([RFC3550], section 6.4.4), the jitter is only meaningful for RTP flows that send a single data packet for each RTP timestamp value (i.e., audio flows, or video flows where each packet comprises one video frame).

3. Receivers report the fraction of RTP data packets lost during the RTCP reporting interval, and the cumulative number of RTP packets lost over the entire RTP session.

These congestion signals limit the possible circuit breakers, since they give only limited visibility into the behaviour of the network.

RTT estimates are widely used in congestion control algorithms, as a proxy for queuing delay measures in delay-based congestion control or to determine connection timeouts. RTT estimates derived from RTCP SR and RR packets sent according to the RTP/AVP timing rules are too infrequent to be useful though, and don't give enough information to distinguish a delay change due to routing updates from queuing delay caused by congestion. Accordingly, we cannot use the RTT estimate alone as an RTP circuit breaker.

Increased jitter can be a signal of transient network congestion, but in the highly aggregated form reported in RTCP RR packets, it offers insufficient information to estimate the extent or persistence of congestion. Jitter reports are a useful early warning of potential network congestion, but provide an insufficiently strong signal to be used as a circuit breaker.

The remaining congestion signals are the packet loss fraction and the cumulative number of packets lost. If considered carefully, these can be effective indicators that congestion is occurring in networks where packet loss is primarily due to queue overflows, although loss caused by non-congestive packet corruption can distort the result in some networks. TCP congestion control [RFC5681] intentionally tries to fill the router queues, and uses the resulting packet loss as congestion feedback. An RTP flow competing with TCP traffic will therefore expect to see a non-zero packet loss fraction that has to be related to TCP dynamics to estimate available capacity. This behaviour of TCP is reflected in the congestion circuit breaker below, and will affect the design of any RTP congestion control protocol.

Two packet loss regimes can be observed: 1) RTCP RR packets show a non-zero packet loss fraction, while the extended highest sequence number received continues to increment; and 2) RR packets show a loss fraction of zero, but the extended highest sequence number received does not increment even though the sender has been transmitting RTP data packets. The former corresponds to the TCP congestion avoidance state, and indicates a congested path that is still delivering data; the latter corresponds to a TCP timeout, and is most likely due to a path failure. A third condition is that data is being sent but no RTCP feedback is received at all, corresponding to a failure of the reverse path. We derive circuit breaker conditions for these loss regimes in the following.

4.1. RTP/AVP Circuit Breaker #1: Media Timeout

If RTP data packets are being sent, but the RTCP SR or RR packets reporting on that SSRC indicate a non-increasing extended highest sequence number received, this is an indication that those RTP data packets are not reaching the receiver. This could be a short-term issue affecting only a few packets, perhaps caused by a slow-to-open firewall or a transient connectivity problem, but if the issue persists, it is a sign of a more ongoing and significant problem. Accordingly, if a sender of RTP data packets receives CB_INTERVAL or more consecutive RTCP SR or RR packets from the same receiver (see Section 4.5), and those packets correspond to its transmission and have a non-increasing extended highest sequence number received field, then that sender SHOULD cease transmission (see Section 4.6). The extended highest sequence number received field is non-increasing if the sender receives at least CB_INTERVAL consecutive RTCP SR or RR packets that report the same value for this field, but it has sent RTP data packets, at a rate of at least one per RTT, that would have caused an increase in the reported value if they had reached the receiver.

The rationale for waiting for CB_INTERVAL or more consecutive RTCP packets with a non-increasing extended highest sequence number is to give enough time for transient reception problems to resolve themselves, but to stop problem flows quickly enough to avoid causing serious ongoing network congestion. A single RTCP report showing no reception could be caused by a transient fault, and so will not cease transmission. Waiting for more than CB_INTERVAL consecutive RTCP reports before stopping a flow might avoid some false positives, but could lead to problematic flows running for a long time period (potentially tens of seconds, depending on the RTCP reporting interval) before being cut off. Equally, an application that sends few packets when the packet loss rate is high runs the risk that the media timeout circuit breaker triggers inadvertently. The chosen timeout interval is a trade-off between these extremes.

The rationale for enforcing a minimum sending rate below which the media timeout circuit breaker will not trigger is to avoid spurious circuit breaker triggers when the number of packets sent per RTCP reporting interval is small (e.g., a telephony application sends only two RTP comfort noise packets during a five second RTCP reporting interval, and both are lost; this is 100% packet loss, but it seems extreme to terminate the RTP session). The one packet per RTT bound derives from [RFC5405].

4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout

In addition to media timeouts, as were discussed in Section 4.1, an RTP session has the possibility of an RTCP timeout. This can occur when RTP data packets are being sent, but there are no RTCP reports returned from the receiver. This is either due to a failure of the receiver to send RTCP reports, or a failure of the return path that is preventing those RTCP reporting from being delivered. In either case, it is not safe to continue transmission, since the sender has no way of knowing if it is causing congestion. Accordingly, an RTP sender that has not received any RTCP SR or RTCP RR packets reporting on the SSRC it is using for three or more of its RTCP reporting intervals SHOULD cease transmission (see Section 4.6). When calculating the timeout, the deterministic RTCP reporting interval, Td, without the randomization factor, and using the fixed minimum interval of Tmin=5 seconds, MUST be used. The rationale for this choice of timeout is as described in Section 6.2 of [RFC3550] ("so that implementations which do not use the reduced value for transmitting RTCP packets are not timed out by other participants prematurely"), as updated by Section 6.1.4 of [I-D.ietf-avtcore-rtp-multi-stream] to account for the use of the RTP /AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124].

To reduce the risk of premature timeout, implementations SHOULD NOT configure the RTCP bandwidth such that Td is larger than 5 seconds. Similarly, implementations that use the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] SHOULD NOT configure T_rr_interval to values larger than 4 seconds (the reduced limit for T_rr_interval follows Section 6.1.3 of [I-D.ietf-avtcore-rtp-multi-stream]).

The choice of three RTCP reporting intervals as the timeout is made following Section 6.3.5 of RFC 3550 [RFC3550]. This specifies that participants in an RTP session will timeout and remove an RTP sender from the list of active RTP senders if no RTP data packets have been received from that RTP sender within the last two RTCP reporting intervals. Using a timeout of three RTCP reporting intervals is therefore large enough that the other participants will have timed out the sender if a network problem stops the data packets it is sending from reaching the receivers, even allowing for loss of some RTCP packets.

If a sender is transmitting a large number of RTP media streams, such that the corresponding RTCP SR or RR packets are too large to fit into the network MTU, the receiver will generate RTCP SR or RR packets in a round-robin manner. In this case, the sender SHOULD treat receipt of an RTCP SR or RR packet corresponding to any SSRC it sent on the same 5-tuple of source and destination IP address, port, and protocol, as an indication that the receiver and return path are working, preventing the RTCP timeout circuit breaker from triggering.

4.3. RTP/AVP Circuit Breaker #3: Congestion

If RTP data packets are being sent, and the corresponding RTCP SR or RR packets show non-zero packet loss fraction and increasing extended highest sequence number received, then those RTP data packets are arriving at the receiver, but some degree of congestion is occurring. The RTP/AVP profile [RFC3551] states that:

> If best-effort service is being used, RTP receivers SHOULD monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path and experiencing the same network conditions would achieve an average throughput, measured on a reasonable time scale, that is not less than the RTP flow is achieving. This condition can be satisfied by implementing congestion control mechanisms to adapt the transmission rate (or the number of layers subscribed for a layered multicast session), or by arranging for a receiver to leave the session if the loss rate is unacceptably high.
>
> The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in time scale and throughput. The time scale on which TCP throughput is measured is the round-trip time of the connection. In essence, this requirement states that it is not acceptable to deploy an application (using RTP or any other transport protocol) on the best-effort Internet which consumes bandwidth arbitrarily and does not compete fairly with TCP within an order of magnitude.

The phase "order of magnitude" in the above means within a factor of ten, approximately. In order to implement this, it is necessary to estimate the throughput a TCP connection would achieve over the path. For a long-lived TCP Reno connection, it has been shown that the TCP throughput can be estimated using the following equation [Padhye]:

$$X = \frac{s}{R*\mathrm{sqrt}(2*b*p/3) + (t\_RTO * (3*\mathrm{sqrt}(3*b*p/8) * p * (1+32*p^2)))}$$

where:

X is the transmit rate in bytes/second.

s is the packet size in bytes. If data packets vary in size, then the average size is to be used.

R is the round trip time in seconds.

p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted.

t_RTO is the TCP retransmission timeout value in seconds, generally approximated by setting t_RTO = 4*R.

b is the number of packets that are acknowledged by a single TCP acknowledgement; [RFC3448] recommends the use of b=1 since many TCP implementations do not use delayed acknowledgements.

This is the same approach to estimated TCP throughput that is used in [RFC3448]. Under conditions of low packet loss the second term on the denominator is small, so this formula can be approximated with reasonable accuracy as follows [Mathis]:

$$X = \frac{s}{R * \mathrm{sqrt}(2*b*p/3)}$$

It is RECOMMENDED that this simplified throughout equation be used, since the reduction in accuracy is small, and it is much simpler to calculate than the full equation. Measurements have shown that the simplified TCP throughput equation is effective as an RTP circuit breaker for multimedia flows sent to hosts on residential networks using ADSL and cable modem links [Singh]. The data shows that the full TCP throughput equation tends to be more sensitive to packet loss and triggers the RTP circuit breaker earlier than the simplified equation. Implementations that desire this extra sensitivity MAY use the full TCP throughput equation in the RTP circuit breaker. Initial measurements in LTE networks have shown that the extra sensitivity is helpful in that environment, with the full TCP throughput equation giving a more balanced circuit breaker response than the simplified TCP equation [Sarker]; other networks might see similar behaviour.

No matter what TCP throughput equation is chosen, two parameters need to be estimated and reported to the sender in order to calculate the throughput: the round trip time, R, and the loss event rate, p (the packet size, s, is known to the sender). The round trip time can be estimated from RTCP SR and RR packets. This is done too infrequently for accurate statistics, but is the best that can be done with the standard RTCP mechanisms.

Report blocks in RTCP SR or RR packets contain the packet loss fraction, rather than the loss event rate, so p cannot be reported (TCP typically treats the loss of multiple packets within a single RTT as one loss event, but RTCP RR packets report the overall fraction of packets lost, and does not report when the packet losses occurred). Using the loss fraction in place of the loss event rate can overestimate the loss. We believe that this overestimate will not be significant, given that we are only interested in order of magnitude comparison ([Floyd] section 3.2.1 shows that the difference is small for steady-state conditions and random loss, but using the loss fraction is more conservative in the case of bursty loss).

The congestion circuit breaker is therefore: when a sender that is transmitting more than one RTP packet per RTT receives an RTCP SR or RR packet that contains a report block for an SSRC it is using, the sender MUST record the value of the fraction lost field in the report block and the time since the last report block was received for that SSRC. If more than CB_INTERVAL (see Section 4.5) report blocks have been received for that SSRC, the sender MUST calculate the average fraction lost over the last CB_INTERVAL reporting intervals, and then estimate the TCP throughput that would be achieved over the path using the chosen TCP throughput equation and the measured values of the round-trip time, R, the loss event rate, p (as approximated by the average fraction lost), and the packet size, s. This estimate of the TCP throughput is then compared with the actual sending rate. If the actual sending rate is more than ten times the TCP throughput estimate, then the congestion circuit breaker is triggered.

The average fraction lost is calculated based on the sum, over the last CB_INTERVAL reporting intervals, of the fraction lost in each reporting interval multiplied by the duration of the corresponding reporting interval, divided by the total duration of the last CB_INTERVAL reporting intervals.

The rationale for enforcing a minimum sending rate below which the congestion circuit breaker will not trigger is to avoid spurious circuit breaker triggers when the number of packets sent per RTCP reporting interval is small, and hence the fraction lost samples are subject to measurement artefacts. The one packet per RTT bound derives from [RFC5405].

When the congestion circuit breaker is triggered, the sender SHOULD cease transmission (see Section 4.6). However, if the sender is able to reduce its sending rate by a factor of (approximately) ten, then it MAY first reduce its sending rate by this factor (or some larger amount) to see if that resolves the congestion. If the sending rate is reduced in this way and the congestion circuit breaker triggers again after the next CB_INTERVAL RTCP reporting intervals, the sender MUST then cease transmission. An example of such a rate reduction might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here).

The RTCP reporting interval of the media sender does not affect how quickly congestion circuit breaker can trigger. The timing is based on the RTCP reporting interval of the receiver that generates the SR/RR packets from which the loss rate and RTT estimate are derived (note that RTCP requires all participants in a session to have similar reporting intervals, else the participant timeout rules in [RFC3550] will not work, so this interval is likely similar to that of the sender). If the incoming RTCP SR or RR packets are using a reduced minimum RTCP reporting interval (as specified in Section 6.2 of RFC 3550 [RFC3550] or the RTP/AVPF profile [RFC4585]), then that reduced RTCP reporting interval is used when determining if the circuit breaker is triggered.

As in Section 4.1 and Section 4.2, we use CB_INTERVAL reporting intervals to avoid triggering the circuit breaker on transient failures. This circuit breaker is a worst-case condition, and congestion control needs to be performed to keep well within this bound. It is expected that the circuit breaker will only be triggered if the usual congestion control fails for some reason.

If there are more media streams that can be reported in a single RTCP SR or RR packet, or if the size of a complete RTCP SR or RR packet exceeds the network MTU, then the receiver will report on a subset of sources in each reporting interval, with the subsets selected round-robin across multiple intervals so that all sources are eventually reported [RFC3550]. When generating such round-robin RTCP reports, priority SHOULD be given to reports on sources that have high packet loss rates, to ensure that senders are aware of network congestion they are causing (this is an update to [RFC3550]).

4.4. RTP/AVP Circuit Breaker #4: Media Usability

Applications that use RTP are generally tolerant to some amount of packet loss. How much packet loss can be tolerated will depend on the application, media codec, and the amount of error correction and packet loss concealment that is applied. There is an upper bound on the amount of loss can be corrected, however, beyond which the media becomes unusable. Similarly, many applications have some upper bound on the media capture to play-out latency that can be tolerated before the application becomes unusable. The latency bound will depend on the application, but typical values can range from the order of a few hundred milliseconds for voice telephony and interactive conferencing applications, up to several seconds for some video-on-demand systems.

As a final circuit breaker, RTP senders SHOULD monitor the reported packet loss and delay to estimate whether the media is likely to be suitable for the intended purpose. If the packet loss rate and/or latency is such that the media has become unusable, and has remained unusable for a significant time period, then the application SHOULD cease transmission. Similarly, receivers SHOULD monitor the quality of the media they receive, and if the quality is unusable for a significant time period, they SHOULD terminate the session. This memo intentionally does not define a bound on the packet loss rate or latency that will result in unusable media, nor does it specify what time period is deemed significant, as these are highly application dependent.

Sending media that suffers from such high packet loss or latency that it is unusable at the receiver is both wasteful of resources, and of no benefit to the user of the application. It also is highly likely to be congesting the network, and disrupting other applications. As such, the congestion circuit breaker will almost certainly trigger to stop flows where the media would be unusable due to high packet loss or latency. However, in pathological scenarios where the congestion circuit breaker does not stop the flow, it is desirable that the RTP application cease sending useless traffic. The role of the media usability circuit breaker is to protect the network in such cases.

4.5. Choice of Circuit Breaker Interval

The CB_INTERVAL parameter determines the number of consecutive RTCP reporting intervals that need to suffer congestion before the media timeout circuit breaker (see Section 4.1) or the congestion circuit breaker (see Section 4.3) triggers. It determines the sensitivity and responsiveness of these circuit breakers.

The CB_INTERVAL parameter is set to min(floor(3+(2.5/Td)), 30) RTCP reporting intervals, where Td is the deterministic calculated RTCP interval described in section 6.3.1 of [RFC3550]. This expression gives an CB_INTERVAL that varies as follows:

Td	CB_INTERVAL	Time to trigger
0.016 seconds	30 RTCP reporting intervals	0.48 seconds
0.033 seconds	30 RTCP reporting intervals	0.99 seconds
0.1 seconds	28 RTCP reporting intervals	2.8 seconds
0.5 seconds	8 RTCP reporting intervals	4.0 seconds
1.0 seconds	5 RTCP reporting intervals	5.5 seconds
2.0 seconds	4 RTCP reporting intervals	8.5 seconds
5.0 seconds	5 RTCP reporting intervals	17.5 seconds
10.0 seconds	3 RTCP reporting intervals	32.5 seconds

If the RTP/AVPF profile [RFC4585] or the RTP/SAVPF [RFC5124] is used, and the T_rr_interval parameter is used to reduce the frequency of regular RTCP reports, then the value Td in the above expression for the CB_INTERVAL parameter MUST be replaced by T_rr_interval.

The CB_INTERVAL parameter is calculated on joining the session, and recalculated on receipt of each RTCP packet, after checking whether the media timeout circuit breaker or the congestion circuit breaker has been triggered.

To ensure a timely response to persistent congestion, implementations SHOULD NOT configure the RTCP bandwidth such that Td is larger than 5 seconds. Similarly, implementations that use the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] SHOULD NOT configure T_rr_interval to values larger than 4 seconds (the reduced limit for T_rr_interval follows Section 6.1.3 of [I-D.ietf-avtcore-rtp-multi-stream]).

Rationale: If the CB_INTERVAL was always set to the same number of RTCP reporting intervals, this would cause higher rate RTP sessions to trigger the RTP circuit breaker after a shorter time interval than lower rate sessions, because the RTCP reporting interval scales based on the RTP session bandwidth. This is felt to penalise high rate RTP sessions too aggressively. Conversely, scaling CB_INTERVAL according to the inverse of the RTCP reporting interval, so the RTP circuit breaker triggers after a constant time interval, doesn't sufficiently protect the network from congestion caused by high-rate flows. The chosen expression for CB_INTERVAL seeks a balance between these two extremes. It causes higher rate RTP sessions subject to persistent congestion to trigger the RTP circuit breaker after a shorter time interval than do lower rate RTP sessions, while also making the RTP circuit breaker for such sessions less sensitive by requiring the congestion to persist for longer numbers of RTCP reporting intervals.

4.6. Ceasing Transmission

What it means to cease transmission depends on the application, but the intention is that the application will stop sending RTP data packets to a particular destination 3-tuple (transport protocol, destination port, IP address), until the user makes an explicit attempt to restart the call. It is important that a human user is involved in the decision to try to restart the call, since that user will eventually give up if the calls repeatedly trigger the circuit breaker. This will help avoid problems with automatic redial systems from congesting the network. Accordingly, RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated.

It is recognised that the RTP implementation in some systems might not be able to determine if a call set-up request was initiated by a human user, or automatically by some scripted higher-level component of the system. These implementations SHOULD rate limit attempts to restart a call to the same destination 3-tuple as used by a previous call that was recently halted by the circuit breaker. The chosen rate limit ought to not exceed the rate at which an annoyed human caller might redial a misbehaving phone.

5. RTP Circuit Breakers and the RTP/AVPF and RTP/SAVPF Profiles

Use of the Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585] allows receivers to send early RTCP reports in some cases, to inform the sender about particular events in the media stream. There are several use cases for such early RTCP reports, including providing rapid feedback to a sender about the onset of congestion. The RTP/SAVPF Profile [RFC5124] is a secure variant of the RTP/AVPF profile, that is treated the same in the context of the RTP circuit breaker. These feedback profiles are often used with non-compound RTCP reports [RFC5506] to reduce the reporting overhead.

Receiving rapid feedback about congestion events potentially allows congestion control algorithms to be more responsive, and to better adapt the media transmission to the limitations of the network. It is expected that many RTP congestion control algorithms will adopt the RTP/AVPF profile or the RTP/SAVPF profile for this reason, defining new transport layer feedback reports that suit their requirements. Since these reports are not yet defined, and likely very specific to the details of the congestion control algorithm chosen, they cannot be used as part of the generic RTP circuit breaker.

Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that do not contain an RTCP SR or RR packet MUST be ignored by the congestion circuit breaker (they do not contain the information needed by the congestion circuit breaker algorithm), but MUST be counted as received packets for the RTCP timeout circuit breaker. Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that contain RTCP SR or RR packets MUST be processed by the congestion circuit breaker as if they were sent as regular RTCP reports, and counted towards the circuit breaker conditions specified in Section 4 of this memo. This will potentially make the RTP circuit breaker trigger earlier than it would if the RTP/AVPF profile was not used.

When using ECN with RTP (see Section 8), early RTCP feedback packets can contain ECN feedback reports. The count of ECN-CE marked packets contained in those ECN feedback reports is counted towards the number of lost packets reported if the ECN Feedback Report report is sent in an compound RTCP packet along with an RTCP SR/RR report packet. Reports of ECN-CE packets sent as reduced-size RTCP ECN feedback packets without an RTCP SR/RR packet MUST be ignored.

These rules are intended to allow the use of low-overhead RTP/AVPF feedback for generic NACK messages without triggering the RTP circuit breaker. This is expected to make such feedback suitable for RTP congestion control algorithms that need to quickly report loss events in between regular RTCP reports. The reaction to reduced-size RTCP SR/RR packets is to allow such algorithms to send feedback that can trigger the circuit breaker, when desired.

The RTP/AVPF and RTP/SAVPF profiles include the T_rr_interval parameter that can be used to adjust the regular RTCP reporting interval. The use of the T_rr_interval parameter changes the behaviour of the RTP circuit breaker, as described in Section 4.

6. Impact of RTCP Extended Reports (XR)

RTCP Extended Report (XR) blocks provide additional reception quality metrics, but do not change the RTCP timing rules. Some of the RTCP XR blocks provide information that might be useful for congestion control purposes, others provided non-congestion-related metrics. With the exception of RTCP XR ECN Summary Reports (see Section 8), the presence of RTCP XR blocks in a compound RTCP packet does not affect the RTP circuit breaker algorithm. For consistency and ease of implementation, only the reception report blocks contained in RTCP SR packets, RTCP RR packets, or RTCP XR ECN Summary Report packets, are used by the RTP circuit breaker algorithm.

7. Impact of RTCP Reporting Groups

An optimisation for grouping RTCP reception statistics and other feedback in RTP sessions with large numbers of participants is given in [I-D.ietf-avtcore-rtp-multi-stream-optimisation]. This allows one SSRC to act as a representative that sends reports on behalf of other SSRCs that are co-located in the same endpoint and see identical reception quality. When running the circuit breaker algorithms, an endpoint MUST treat a reception report from the representative of the reporting group as if a reception report was received from all members of that group.

8. Impact of Explicit Congestion Notification (ECN)

The use of ECN for RTP flows does not affect the media timeout RTP circuit breaker (Section 4.1) or the RTCP timeout circuit breaker (Section 4.2), since these are both connectivity checks that simply determinate if any packets are being received.

ECN-CE marked packets SHOULD be treated as if it were lost for the purposes of congestion control, when determining the optimal media sending rate for an RTP flow. If an RTP sender has negotiated ECN support for an RTP session, and has successfully initiated ECN use on the path to the receiver [RFC6679], then ECN-CE marked packets SHOULD be treated as if they were lost when calculating if the congestion-based RTP circuit breaker (Section 4.3) has been met. The count of ECN-CE marked RTP packets is returned in RTCP XR ECN summary report packets if support for ECN has been initiated for an RTP session.

9. Impact of Bundled Media and Layered Coding

The RTP circuit breaker operates on a per-RTP session basis. An RTP sender that participates in several RTP sessions MUST treat each RTP session independently with regards to the RTP circuit breaker.

An RTP sender can generate several media streams within a single RTP session, with each stream using a different SSRC. This can happen if bundled media are in use, when using simulcast, or when using layered media coding. By default, each SSRC will be treated independently by the RTP circuit breaker. However, the sender MAY choose to treat the flows (or a subset thereof) as a group, such that a circuit breaker trigger for one flow applies to the group of flows as a whole, and either causes the entire group to cease transmission, or the sending rate of the group to reduce by a factor of ten, depending on the RTP circuit breaker triggered. Grouping flows in this way is expected to be especially useful for layered flows sent using multiple SSRCs, as it allows the layered flow to react as a whole, ceasing transmission on the enhancement layers first to reduce sending rate if necessary, rather than treating each layer independently.

10. Security Considerations

The security considerations of [RFC3550] apply.

If the RTP/AVPF profile is used to provide rapid RTCP feedback, the security considerations of [RFC4585] apply. If ECN feedback for RTP over UDP/IP is used, the security considerations of [RFC6679] apply.

If non-authenticated RTCP reports are used, an on-path attacker can trivially generate fake RTCP packets that indicate high packet loss rates, causing the circuit breaker to trigger and disrupting an RTP session. This is somewhat more difficult for an off-path attacker, due to the need to guess the randomly chosen RTP SSRC value and the RTP sequence number. This attack can be avoided if RTCP packets are authenticated; authentication options are discussed in [RFC7201].

Timely operation of the RTP circuit breaker depends on the choice of RTCP reporting interval. If the receiver has a reporting interval that is overly long, then the responsiveness of the circuit breaker decreases. In the limit, the RTP circuit breaker can be disabled for all practical purposes by configuring an RTCP reporting interval that is many minutes duration. This issue is not specific to the circuit breaker: long RTCP reporting intervals also prevent reception quality reports, feedback messages, codec control messages, etc., from being used. Implementations are expected to impose an upper limit on the RTCP reporting interval they are willing to negotiate (based on the session bandwidth and RTCP bandwidth fraction) when using the RTP circuit breaker, as discussed in Section 4.5.

11. IANA Considerations

There are no actions for IANA.

12. Acknowledgements

The authors would like to thank Bernard Aboba, Harald Alvestrand, Gorry Fairhurst, Nazila Fough, Kevin Gross, Cullen Jennings, Randell Jesup, Jonathan Lennox, Matt Mathis, Stephen McQuistin, Eric Rescorla, Abheek Saha, Fabio Verdicchio, and Magnus Westerlund for their valuable feedback.

13. References

13.1. Normative References

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

[RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

[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.

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

[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.

13.2. Informative References

[Floyd] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", Proceedings of the ACM SIGCOMM conference, 2000, DOI 10.1145/347059.347397, August 2000.

[I-D.ietf-avtcore-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-05 (work in progress), February 2015.

[I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-06 (work in progress), October 2014.

[Mathis] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm", ACM SIGCOMM Computer Communication Review 27(3), DOI 10.1145/263932.264023, July 1997.

[Padhye] Padhye, J., Firoiu, V., Towsley, D., and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proceedings of the ACM SIGCOMM conference, 1998, DOI 10.1145/285237.285291, August 1998.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

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

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

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

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

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

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.

[RFC6798] Clark, A. and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", RFC 6798, November 2012.

[RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, January 2013.

[RFC6958] Clark, A., Zhang, S., Zhao, J., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss Metric Reporting", RFC 6958, May 2013.

[RFC7002] Clark, A., Zorn, G., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count Metric Reporting", RFC 7002, September 2013.

[RFC7003] Clark, A., Huang, R., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, September 2013.

[RFC7097] Ott, J., Singh, V., and I. Curcio, "RTP Control Protocol (RTCP) Extended Report (XR) for RLE of Discarded Packets", RFC 7097, January 2014.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

[Sarker] Sarker, Z., Singh, V., and C.S. Perkins, "An Evaluation of RTP Circuit Breaker Performance on LTE Networks", Proceedings of the IEEE Infocom workshop on Communication and Networking Techniques for Contemporary Video, 2014, April 2014.

[Singh] Singh, V., McQuistin, S., Ellis, M., and C.S. Perkins, "Circuit Breakers for Multimedia Congestion Control", Proceedings of the International Packet Video Workshop, 2013, DOI 10.1109/PV.2013.6691439, December 2013.

Authors' Addresses

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org

Varun Singh
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: varun@comnet.tkk.fi
URI: http://www.netlab.tkk.fi/˜varun/

AVTCORE
Internet-Draft
Updates: 3550, 4585 (if approved)
Intended status: Standards Track
Expires: September 10, 2015

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
March 9, 2015

Sending Multiple Media Streams in a Single RTP Session
draft-ietf-avtcore-rtp-multi-stream-07

Abstract

This memo expands and clarifies the behaviour of Real-time Transport Protocol (RTP) endpoints that use multiple synchronization sources (SSRCs). This occurs, for example, when an endpoint sends multiple media streams in a single RTP session. This memo updates RFC 3550 with regards to handling multiple SSRCs per endpoint in RTP sessions, with a particular focus on RTCP behaviour. It also updates RFC 4585 to update and clarify the calculation of the timeout of SSRCs and the inclusion of feedback messages.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

At the time the Real-Time Transport Protocol (RTP) [RFC3550] was originally designed, and for quite some time after, endpoints in RTP sessions typically only transmitted a single media stream, and thus used a single synchronization source (SSRC) per RTP session, where separate RTP sessions were typically used for each distinct media type. Recently, however, a number of scenarios have emerged in which endpoints wish to send multiple RTP media streams, distinguished by distinct RTP synchronization source (SSRC) identifiers, in a single RTP session. These are outlined in Section 3. Although the initial design of RTP did consider such scenarios, the specification was not consistently written with such use cases in mind. The specifications are thus somewhat unclear.

This memo updates [RFC3550] to clarify behaviour in use cases where endpoints use multiple SSRCs. It also updates [RFC4585] in regards to the timeout of inactive SSRCs to resolve problematic behaviour as well as clarifying the inclusion of feedback messages.

2. Terminology

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 RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Use Cases For Multi-Stream Endpoints

This section discusses several use cases that have motivated the development of endpoints that sends RTP data using multiple SSRCs in a single RTP session.

3.1. Endpoints with Multiple Capture Devices

The most straightforward motivation for an endpoint to send multiple simultaneous RTP streams in a session is the scenario where an endpoint has multiple capture devices, and thus media sources, of the same media type and characteristics. For example, telepresence endpoints, of the type described by the CLUE Telepresence Framework [I-D.ietf-clue-framework], often have multiple cameras or microphones covering various areas of a room, and hence send several RTP streams.

3.2. Multiple Media Types in a Single RTP Session

Recent work has updated RTP [I-D.ietf-avtcore-multi-media-rtp-session] and SDP [I-D.ietf-mmusic-sdp-bundle-negotiation] to remove the historical assumption in RTP that media sources of different media types would always be sent on different RTP sessions. In this work, a single endpoint's audio and video RTP media streams (for example) are instead sent in a single RTP session to reduce the number of transport layer flows used.

3.3. Multiple Stream Mixers

There are several RTP topologies which can involve a central device that itself generates multiple RTP media streams in a session. An example is a mixer providing centralized compositing for a multi-capture scenario like that described in Section 3.1. In this case, the centralized node is behaving much like a multi-capturer endpoint, generating several similar and related sources.

A more complex example is the selective forwarding middlebox, described in Section 3.7 of [I-D.ietf-avtcore-rtp-topologies-update]. This is a middlebox that receives media streams from several endpoints, and then selectively forwards modified versions of some RTP streams toward the other endpoints to which it is connected. For each connected endpoint, a separate media source appears in the session for every other source connected to the middlebox, "projected" from the original streams, but at any given time many of them can appear to be inactive (and thus are receivers, not senders, in RTP). This sort of device is closer to being an RTP mixer than an RTP translator, in that it terminates RTCP reporting about the mixed streams, and it can re-write SSRCs, timestamps, and sequence numbers, as well as the contents of the RTP payloads, and can turn sources on and off at will without appearing to be generating packet loss. Each projected stream will typically preserve its original RTCP source description (SDES) information.

3.4. Multiple SSRCs for a Single Media Source

There are also several cases where a single media source results in the usage of multiple SSRCs within the same RTP session. Transport robustness tools like RTP Retransmission [RFC4588] result in multiple SSRCs, one with source data, and another with the repair data. Scalable encoders and their RTP payload formats, like H.264's extension for Scalable Video Coding(SVC) [RFC6190] can be transmitted in a configuration where the scalable layers are distributed over multiple SSRCs within the same session, to enable RTP packet stream level (SSRC) selection and routing in conferencing middleboxes.

4. Use of RTP by endpoints that send multiple media streams

Every RTP endpoint will have an allocated share of the available session bandwidth, as determined by signalling and congestion control. The endpoint MUST keep its total media sending rate within this share. However, endpoints that send multiple media streams do not necessarily need to subdivide their share of the available bandwidth independently or uniformly to each media stream and its SSRCs. In particular, an endpoint can vary the allocation to different streams depending on their needs, and can dynamically change the bandwidth allocated to different SSRCs (for example, by using a variable rate codec), provided the total sending rate does not exceed its allocated share. This includes enabling or disabling media streams and their redundancy streams as more or less bandwidth becomes available.

5. Use of RTCP by Endpoints that send multiple media streams

The RTP Control Protocol (RTCP) is defined in Section 6 of [RFC3550]. The description of the protocol is phrased in terms of the behaviour of "participants" in an RTP session, under the assumption that each endpoint is a participant with a single SSRC. However, for correct operation in cases where endpoints can send multiple media streams, the specification needs to be interpreted with each SSRC counting as a participant in the session, so that an endpoint that has multiple SSRCs counts as multiple participants. The following describes several concrete cases where this applies.

5.1. RTCP Reporting Requirement

An RTP endpoint that has multiple SSRCs MUST treat each SSRC as a separate participant in the RTP session, sending RTCP reports for each of its SSRCs in every RTCP reporting interval. If the mechanism in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] is not used, then each SSRC will send RTCP reports for all other SSRCs, including those co-located at the same endpoint.

If the endpoint has some SSRCs that are sending data and some that are only receivers, then they will receive different shares of the RTCP bandwidth and calculate different base RTCP reporting intervals. Otherwise, all SSRCs at an endpoint will calculate the same base RTCP reporting interval. The actual reporting intervals for each SSRC are randomised in the usual way, but reports can be aggregated as described in Section 5.3.

5.2. Initial Reporting Interval

When a participant joins a unicast session, the following text from Section 6.2 of [RFC3550] is relevant: "For unicast sessions... the delay before sending the initial compound RTCP packet MAY be zero." The basic assumption is that this also ought to apply in the case of multiple SSRCs. Caution has to be exercised, however, when an endpoint (or middlebox) with a large number of SSRCs joins a unicast session, since immediate transmission of many RTCP reports can create a significant burst of traffic, leading to transient congestion and packet loss due to queue overflows.

To ensure that the initial burst of traffic generated by an RTP endpoint is no larger than would be generated by a TCP connection, an RTP endpoint MUST NOT send more than four compound RTCP packets with zero initial delay when it joins a session. Each of those initial compound RTCP packets MAY include aggregated reports from multiple SSRCs, provided the total compound RTCP packet size does not exceed the MTU, and the avg_rtcp_packet_size is maintained as in Section 5.3.1. Aggregating reports from several SSRCs in the initial compound RTCP packets allows a substantial number of SSRCs to report immediately. Endpoints SHOULD prioritize reports on SSRCs that are likely to be most immediately useful, e.g., for SSRCs that are initially senders.

An endpoint that needs to report on more SSRCs than will fit into the four compound RTCP reports that can be sent immediately MUST send the other reports later, following the usual RTCP timing rules including timer reconsideration. Those reports MAY be aggregated as described in Section 5.3.

> Note: The above is based on an TCP initial window of 4 packets, not the larger initial windows which there is an ongoing experiment with. The reason for this is a desire to be conservative as an RTP endpoint will also in many cases commence RTP transmission at the same time as these initial RTCP packets are sent.

5.3. Aggregation of Reports into Compound RTCP Packets

As outlined in Section 5.1, an endpoint with multiple SSRCs has to treat each SSRC as a separate participant when it comes to sending RTCP reports. This will lead to each SSRC sending a compound RTCP packet in each reporting interval. Since these packets are coming from the same endpoint, it might reasonably be expected that they can be aggregated to reduce overheads. Indeed, Section 6.1 of [RFC3550] allows RTP translators and mixers to aggregate packets in similar circumstances:

> "It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7). An example RTCP compound packet as might be produced by a mixer is shown in Fig. 1. If the overall length of a compound packet would exceed the MTU of the network path, it SHOULD be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. This does not impair the RTCP bandwidth estimation because each compound packet represents at least one distinct participant. Note that each of the compound packets MUST begin with an SR or RR packet."

This allows RTP translators and mixers to generate compound RTCP packets that contain multiple SR or RR packets from different SSRCs, as well as any of the other packet types. There are no restrictions on the order in which the RTCP packets can occur within the compound packet, except the regular rule that the compound RTCP packet starts with an SR or RR packet. Due to this rule, correctly implemented RTP endpoints will be able to handle compound RTCP packets that contain RTCP packets relating to multiple SSRCs.

Accordingly, endpoints that use multiple SSRCs MAY aggregate the RTCP packets sent by their different SSRCs into compound RTCP packets, provided 1) the resulting compound RTCP packets begin with an SR or RR packet; 2) they maintain the average RTCP packet size as described in Section 5.3.1; and 3) they schedule packet transmission and manage aggregation as described in Section 5.3.2.

5.3.1. Maintaining AVG_RTCP_SIZE

The RTCP scheduling algorithm in [RFC3550] works on a per-SSRC basis. Each SSRC sends a single compound RTCP packet in each RTCP reporting interval. When an endpoint uses multiple SSRCs, it is desirable to aggregate the compound RTCP packets sent by its SSRCs, reducing the overhead by forming a larger compound RTCP packet. This aggregation can be done as described in Section 5.3.2, provided the average RTCP packet size calculation is updated as follows.

Participants in an RTP session update their estimate of the average RTCP packet size (avg_rtcp_size) each time they send or receive an RTCP packet (see Section 6.3.3 of [RFC3550]). When a compound RTCP packet that contains RTCP packets from several SSRCs is sent or received, the avg_rtcp_size estimate for each SSRC that is reported upon is updated using div_packet_size rather than the actual packet size:

```
avg_rtcp_size = (1/16) * div_packet_size + (15/16) * avg_rtcp_size
```

where div_packet_size is packet_size divided by the number of SSRCs reporting in that compound packet. The number of SSRCs reporting in a compound packet is determined by counting the number of different SSRCs that are the source of Sender Report (SR) or Receiver Report (RR) RTCP packets within the compound RTCP packet. Non-compound RTCP packets (i.e., RTCP packets that do not contain an SR or RR packet [RFC5506]) are considered to report on a single SSRC.

An SSRC that doesn't follow the above rule, and instead uses the full RTCP compound packet size to calculate avg_rtcp_size, will derive an RTCP reporting interval that is overly large by a factor that is proportional to the number of SSRCs aggregated into compound RTCP packets and the size of set of SSRCs being aggregated relative to the total number of participants. This increased RTCP reporting interval can cause premature timeouts if it is more than five times the interval chosen by the SSRCs that understand compound RTCP that aggregate reports from many SSRCs. A 1500 octet MTU can fit five typical size reports into a compound RTCP packet, so this is a real concern if endpoints aggregate RTCP reports from multiple SSRCs.

The issue raised in the previous paragraph is mitigated by the modification in timeout behaviour specified in Section 6.1.2. This mitigation is in place in those cases where the RTCP bandwidth is sufficiently high that an endpoint, using an avg_rtcp_size calculated without taking into account the number of reporting SSRCs, can transmit more frequently than approximately every 5 seconds. Note, however, that the non-modified endpoint's RTCP reporting is still negatively impacted even if the premature timeout of its SSRCs are avoided. If compatibility with non-updated endpoints is a concern, the number of reports from different SSRCs aggregated into a single compound RTCP packet SHOULD either be limited to two reports, or aggregation ought not used at all. This will limit the non-updated endpoint's RTCP reporting interval to be no larger than twice the RTCP reporting interval that would be chosen by an endpoint following this specification.

5.3.2. Scheduling RTCP with Multiple Reporting SSRCs

When implementing RTCP packet scheduling for cases where multiple reporting SSRCs are aggregating their RTCP packets in the same compound packet there are a number of challenges. First of all, we have the goal of not changing the general properties of the RTCP packet transmissions, which include the general inter-packet distribution, and the behaviour for dealing with flash joins as well as other dynamic events.

The below specified mechanism deals with:

o That one can't have a-priori knowledge about which RTCP packets are to be sent, or their size, prior to generating the packets. In which case, the time from generation to transmission ought to be as short as possible to minimize the information that becomes stale.

o That one has an MTU limit, that one ought to avoid exceeding, as that requires lower-layer fragmentation (e.g., IP fragmentation) which impacts the packets' probability of reaching the receiver(s).

The below text modifies and extends the behavior defined in Section 6.3 of [RFC3550], and in Section 3.5.3 of [RFC4585] if the AVPF or SAVPF profile is used, regarding actions to take when scheduling and sending an RTCP packet. It uses the variable names tn, tp, tc, T and Td defined in Section 6.3 of [RFC3550]. The variable T_rr_last is defined in [RFC4585].

Schedule all the endpoint's local SSRCs individually for transmission using the regular calculation of tn for the profile being used. Each time an SSRC's tn timer expires, do the regular reconsideration and, if applicable, T_rr_int based suppression. If the result indicates that an RTCP packet is to be sent and the transmission is a regular RTCP packet:

1. Consider if an additional SSRC can be added. That consideration is done by picking the SSRC which has the tn value closest in time to the current time (tc).

2. Calculate how much space for RTCP packets would be needed to add that SSRC.

3. If the considered SSRC's RTCP Packets fit within the lower layer datagram's Maximum Transmission Unit, taking the necessary protocol headers and the space consumed by prior SSRCs into account, then add that SSRC's RTCP packets to the compound packet and go again to Step 1.

4. Otherwise, if the considered SSRC's RTCP Packets will not fit within the compound packet, then transmit the generated compound packet.

5. Update the RTCP Parameters for each SSRC that has been included in the sent RTCP packet. The previous RTCP transmit time (tp) value for each SSRC MUST be updated as follows:

   A. For the first SSRC set the transmission time (tt) to tc.

B. For any additional SSRC calculate the transmission time that each of these SSRCs would have had it not been aggregated and given the current existing session context. This value is derived by taking this SSRC's tn value and performing reconsideration and updating tn until tp + T <= tn, then set tt = tn. If AVPF or SAVPF is being used, then T_rr_int based suppression MUST NOT be used in this calcualtion.

C. Calculate average transmission time (tt_avg) using the tt of all the SSRCs included in the packet.

D. Now update tp for all the sent SSRCs to tt_avg.

E. If AVPF or SAVPF profile is being used update T_rr_last to tt_avg.

6. For the sent SSRCs calculate new tn values based on the updated parameters and reschedule the timers.

When using AVPF or SAVPF profile, when following the scheduling algorithm for regular transmission in Section 3.5.3 then the case of T_rr_interval == 0, as well as option 1, 2a and 2b for T_rr_interval != 0, results in transmission of a regular RTCP packet that follows the above and updates the necessary variables. However, when the transmission is suppressed per 2c, then tp is updated to tc, as no aggregation has taken place.

Reverse reconsideration needs to be performed as specified in RTP [RFC3550]. It is important to note that under the above algorithm when performing reconsideration, the value of tp can actually be larger than tc. However, that still has the desired effect of proportionally pulling the tp value towards tc (as well as tn) as the group size shrinks in direct proportion the reduced group size.

The above algorithm has been shown in simulations to maintain the inter-RTCP-packet transmission distribution for the SSRCs and consume the same amount of bandwidth as non-aggregated packets in RTP sessions. With this algorithm the actual transmission interval for any SSRC triggering an RTCP compound packet transmission is following the regular transmission rules. The value tp is set to somewhere in the interval [0,1.5/1.21828*Td] ahead of tc. The actual value is average of one instance of tc and the randomized transmission times of the additional SSRCs, thus the lower range of the interval is more probable. This setting is performed to compensate for the bias that is otherwise introduced by picking the shortest tn value out of the N SSRCs included in aggregate.

The algorithm also handles the cases where the number of SSRCs that can be included in an aggregated packet varies. An SSRC that previously was aggregated and fails to fit in a packet still has its own transmission scheduled according to normal rules. Thus, it will trigger a transmission in due time, or the SSRC will be included in another aggregate. The algorithm's behaviour under SSRC group size changes is as follows:

RTP sessions where the number of SSRC are growing: When the group size is growing, the Td values grow in proportion to the number of new SSRCs in the group. When reconsideration is done when the timer for the tn expires, that SSRC will reconsider the transmission and with a certain probability reschedule the tn timer. This part of the reconsideration algorithm is only impacted by the above algorithm by having tp values that were in the future instead of set to the time of the actual last transmission at the time of updating tp.

RTP sessions where the number of SSRC are shrinking: When the group shrinks, reverse reconsideration moves the tp and tn values towards tc proportionally to the number of SSRCs that leave the session compared to the total number of participants when they left. The setting of the tp value forward in time related to the tc could be believed to have negative effect. However, the reason for this setting is to compensate for bias caused by picking the shortest tn out of the N aggregated. This bias remains over a reduction in the number of SSRCs. The reverse reconsideration compensates the reduction independently of aggregation being used or not. The negative effect that can occur on removing an SSRC is that the most favourable tn belonged to the removed SSRC. The impact of this is limited to delaying the transmission, in the worst case, one reporting interval.

In conclusion the investigations performed has found no significant negative impact on the scheduling algorithm.

5.4. Use of RTP/AVPF Feedback

This section discusses the transmission of RTP/AVPF feedback packets when the transmitting endpoint has multiple SSRCs.

5.4.1. Choice of SSRC for Feedback Packets

When an RTP/AVPF endpoint has multiple SSRCs, it can choose what SSRC to use as the source for the RTCP feedback packets it sends. Several factors can affect that choice:

- o RTCP feedback packets relating to a particular media type SHOULD be sent by an SSRC that receives that media type. For example, when audio and video are multiplexed onto a single RTP session, endpoints will use their audio SSRC to send feedback on the audio received from other participants.

- o RTCP feedback packets and RTCP codec control messages that are notifications or indications regarding RTP data processed by an endpoint MUST be sent from the SSRC used by that RTP data. This includes notifications that relate to a previously received request or command [RFC4585][RFC5104].

- o If separate SSRCs are used to send and receive media, then the corresponding SSRC SHOULD be used for feedback, since they have differing RTCP bandwidth fractions. This can also affect the consideration if the SSRC can be used in immediate mode or not.

- o Some RTCP feedback packet types require consistency in the SSRC used. For example, if a TMMBR limitation [RFC5104] is set by an SSRC, the same SSRC needs to be used to remove the limitation.

- o If several SSRCs are suitable for sending feedback, if might be desirable to use an SSRC that allows the sending of feedback as an early RTCP packet.

When an RTCP feedback packet is sent as part of a compound RTCP packet that aggregates reports from multiple SSRCs, there is no requirement that the compound packet contains an SR or RR packet generated by the sender of the RTCP feedback packet. For reduced-size RTCP packets, aggregation of RTCP feedback packets from multiple sources is not limited further than Section 4.2.2 of [RFC5506].

5.4.2. Scheduling an RTCP Feedback Packet

When an SSRC has a need to transmit a feedback packet in early mode it follows the scheduling rules defined in Section 3.5 in RTP/AVPF [RFC4585]. When following these rules the following clarifications need to be taken into account:

- o Whether a session is considered to be point-to-point or multiparty is not based on the number of SSRCs, but the number of endpoints one directly interacts with in the RTP session. This is determined by counting the number of CNAMEs used by the SSRCs received. A RTP session MUST be considered multiparty if more than one CNAME is received, unless signalling explicitly indicates that the session is to be handled as point to point, or RTCP reporting groups [I-D.ietf-avtcore-rtp-multi-stream-optimisation] are used. If RTCP reporting groups are used, the classification is solely based on whether the endpoint receives a single reporting group, indicating point to point, or if multiple reporting groups are received (or a mixture of sources using and sources not using reporting groups), which is classified as multiparty. Note that contributing sources (CSRCs) can be bound to any number of different CNAMEs and do not affect the determination of whether the session is multiparty. Similarly, SSRC/CSRC values that are only seen in the source field of an SDES packet do not affect this determination.

- o Note that when checking if there is already a scheduled compound RTCP packet containing feedback messages (Step 2 in Section 3.5.2), that check is done considering all local SSRCs.

- o If the SSRC is not allowed to send an early RTCP packet, then the feedback message MAY be queued for transmission as part of any early or regular scheduled transmission that can occur within the maximum useful lifetime of the feedback message (T_max_fb_delay). This modifies the behaviour in bullet 4a) in Section 3.5.2 of [RFC4585].

The above rule for determining if a RTP session is to be considered point-to-point or multiparty is simple and straightforward and works in most cases. The goal with the above classification is to determine if the resources associated with RTP and RTCP are shared with only one peer or multiple other endpoints. This is significant as it affects the impact and the necessary processing and resource consumption. Relying on only CNAME will result in classifying some few situations where one might actually have only one peer as a multiparty situation. The known situations are the following ones:

Endpoint with multiple synchronization contexts: An endpoint that is part of a point-to-point session can have multiple synchronization contexts, for example due to forwarding an external media source into a interactive real-time conversation. In this case the classification will consider the peer as two endpoints, while the actual RTP/RTCP transmission will be under the control of one endpoint.

Selective Forwarding Middlebox: The SFM as defined in Section 3.7 of [I-D.ietf-avtcore-rtp-topologies-update] has control over the transmission and configurations between itself and each peer endpoint individually. It also fully controls the RTCP packets being forwarded between the individual legs. Thus, this type of middlebox can be compared to the RTP mixer, which uses its own SSRCs to mix or select the media it forwards, that will be classified as a point-to-point RTP session by the above rule.

In the above cases it is very reasonable to use RTCP reporting groups [I-D.ietf-avtcore-rtp-multi-stream-optimisation]. If that extension is used, an endpoint can indicate that the multitude of CNAMEs are in fact under a single endpoint or middlebox control by using only a single reporting group.

The above rules will also classify some sessions where the endpoint is connected to an RTP mixer as being point to point. For example the mixer could act as gateway to an Any Source Multicast based RTP session for the discussed endpoint. However, this will in most cases be okay, as the RTP mixer provides separation between the two parts of the session. The responsibility falls on the mixer to act accordingly in each domain.

Note: The above usage of point-to-point or multiparty as classifiers is actually misleading, but we maintain these labels to match what is used in [RFC4585] as this ensures that the right algorithms are applied.

To conclude we note that in some cases signalling can be used to override the rule when it would result in the wrong classification.

6. RTCP Considerations for Streams with Disparate Rates

An RTP session has a single set of parameters that configure the session bandwidth. These are the RTCP sender and receiver fractions (e.g., the SDP "b=RR:" and "b=RS:" lines), and the parameters of the RTP/AVPF profile [RFC4585] (e.g., trr-int) if that profile (or its secure extension, RTP/SAVPF [RFC5124]) is used. As a consequence, the base RTCP reporting interval, before randomisation, will be the same for every sending SSRC in an RTP session. Similarly, every receiving SSRC in an RTP session will have the same base reporting interval, although this can differ from the reporting interval chosen by sending SSRCs. This uniform RTCP reporting interval for all SSRCs can result in RTCP reports being sent more often, or too seldom, than is considered desirable for a RTP stream.

For example, consider a scenario when an audio flow sending at tens of kilobits per second is multiplexed into an RTP session with a multi-megabit high quality video flow. If the session bandwidth is configured based on the video sending rate, and the default RTCP bandwidth fraction of 5% of the session bandwidth is used, it is likely that the RTCP bandwidth will exceed the audio sending rate. If the reduced minimum RTCP interval described in Section 6.2 of [RFC3550] is then used in the session, as appropriate for video where rapid feedback on damaged I-frames is wanted, the uniform reporting interval for all senders could mean that audio sources are expected to send RTCP packets more often than they send audio data packets.

This bandwidth mismatch can be reduced by careful tuning of the RTCP parameters, especially trr_int when the RTP/AVPF profile is used, cannot be avoided entirely, as it is inherent in the design of the RTCP timing rules, and affects all RTP sessions that contain flows with greatly mismatched bandwidth.

Different media rates or desired RTCP behaviours can also occur between SSRCs carrying the same media type. A common case in multiparty conferencing is when only one or two video source are shown in higher resolution, while the others are shown as small thumbnails, with the choice of which is shown in high resolution being voice activity controlled. Here the differences are both in actual media rate and in choices for what feedback messages might be needed. Other examples of differences that can exist are due to the intended usage of a media source. A media source carrying the video of the speaker in a conference is different from a document camera. Basic parameters that can differ in this case are frame-rate, acceptable end-to-end delay, and the SNR fidelity of the image. These differences affect not only the needed bit-rates, but also possible transmission behaviours, usable repair mechanisms, what feedback messages the control and repair requires, the transmission requirements on those feedback messages, and monitoring of the RTP stream delivery.

Sending multiple media types in a single RTP session causes that session to contain more SSRCs than if each media type was sent in a separate RTP session. For example, if two participants each send an audio and a video flow in a single RTP session, that session will comprise four SSRCs, but if separate RTP sessions had been used for audio and video, each of those two RTP sessions would comprise only two SSRCs. Sending multiple media streams in an RTP session hence increases the amount of cross reporting between the SSRCs, as each SSRC reports on all other SSRCs in the session. This increases the size of the RTCP reports, causing them to be sent less often than would be the case if separate RTP sessions where used for a given RTCP bandwidth.

Finally, when an RTP session contains multiple media types, it is important to note that the RTCP reception quality reports, feedback messages, and extended report blocks used might not be applicable to all media types. Endpoints will need to consider the media type of each SSRC only send or process reports and feedback that apply to that particular SSRC and its media type. Signalling solutions might have shortcomings when it comes to indicating that a particular set of RTCP reports or feedback messages only apply to a particular media type within an RTP session.

From an RTCP perspective, therefore, it can be seen that there are advantages to using separate RTP sessions for each media stream, rather than sending multiple media streams in a single RTP session. However, these are frequently offset by the need to reduce port use, to ease NAT/firewall traversal, achieved by combining media streams into a single RTP session. The following sections consider some of the issues with using RTCP in sessions with multiple media streams in more detail.

6.1. Timing out SSRCs

Various issues have been identified with timing out SSRC values when sending multiple media streams in an RTP session.

6.1.1. Problems with RTP/AVPF the T_rr_interval Parameter

The RTP/AVPF profile includes a method to prevent RTCP reports from being sent too often. This mechanism is described in Section 3.5.3 of [RFC4585], and is controlled by the T_rr_interval parameter. It works as follows. When a regular RTCP report is sent, a new random value, T_rr_current_interval, is generated, drawn evenly in the range 0.5 to 1.5 times T_rr_interval. If a regular RTCP packet is to be sent earlier then T_rr_current_interval seconds after the previous regular RTCP packet, and there are no feedback messages to be sent, then that regular RTCP packet is suppressed, and the next regular RTCP packet is scheduled. The T_rr_current_interval is recalculated each time a regular RTCP packet is sent. The benefit of suppression is that it avoids wasting bandwidth when there is nothing requiring frequent RTCP transmissions, but still allows utilization of the configured bandwidth when feedback is needed.

Unfortunately this suppression mechanism skews the distribution of the RTCP sending intervals compared to the regular RTCP reporting intervals. The standard RTCP timing rules, including reconsideration and the compensation factor, result in the intervals between sending RTCP packets having a distribution that is skewed towards the upper end of the range [0.5/1.21828, 1.5/1.21828]*Td, where Td is the deterministic calculated RTCP reporting interval. With Td = 5s this distribution covers the range [2.052s, 6.156s]. In comparison, the RTP/AVPF suppression rules act in an interval that is 0.5 to 1.5 times T_rr_interval; for T_rr_interval = 5s this is [2.5s, 7.5s].

The effect of this is that the time between consecutive RTCP packets when using T_rr_interval suppression can become large. The maximum time interval between sending one regular RTCP packet and the next, when T_rr_interval is being used, occurs when T_rr_current_interval takes its maximum value and a regular RTCP packet is suppressed at the end of the suppression period, then the next regular RTCP packet is scheduled after its largest possible reporting interval. Taking the worst case of the two intervals gives a maximum time between two RTCP reports of 1.5*T_rr_interval + 1.5/1.21828*Td.

This behaviour can be surprising when Td and T_rr_interval have the same value. That is, when T_rr_interval is configured to match the regular RTCP reporting interval. In this case, one might expect that regular RTCP packets are sent according to their usual schedule, but feedback packets can be sent early. However, the above-mentioned issue results in the RTCP packets actually being sent in the range [0.5*Td, 2.731*Td] with a highly non-uniform distribution, rather than the range [0.41*Td, 1.23*Td]. This is perhaps unexpected, but is not a problem in itself. However, when coupled with packet loss, it raises the issue of premature timeout.

6.1.2. Avoiding Premature Timeout

In RTP/AVP [RFC3550] the timeout behaviour is simple, and is 5 times Td, where Td is calculated with a Tmin value of 5 seconds. In other words, if the configured RTCP bandwidth allows for an average RTCP reporting interval shorter than 5 seconds, the timeout is 25 seconds of no activity from the SSRC (RTP or RTCP), otherwise the timeout is 5 average reporting intervals.

RTP/AVPF [RFC4585] introduces different timeout behaviours depending on the value of T_rr_interval. When T_rr_interval is 0, it uses the same timeout calculation as RTP/AVP. However, when T_rr_interval is non-zero, it replaces Tmin in the timeout calculation, most likely to speed up detection of timed out SSRCs. However, using a non-zero T_rr_interval has two consequences for RTP behaviour.

First, due to suppression, the number of RTP and RTCP packets sent by an SSRC that is not an active RTP sender can become very low, because of the issue discussed in Section 6.1.1. As the RTCP packet interval can be as long as 2.73*Td, then during a 5*Td time period an endpoint might in fact transmit only a single RTCP packet. The long intervals result in fewer RTCP packets, to a point where a single RTCP packet loss can sometimes result in timing out an SSRC.

Second, the RTP/AVPF changes to the timeout rules reduce robustness to misconfiguration. It is common to use RTP/AVPF configured such that RTCP packets can be sent frequently, to allow rapid feedback, however this makes timeouts very sensitive to T_rr_interval. For example, if two SSRCs are configured one with T_rr_interval = 0.1s and the other with T_rr_interval = 0.6s, then this small difference will result in the SSRC with the shorter T_rr_interval timing out the other if it stops sending RTP packets, since the other RTCP reporting interval is more than five times its own. When RTP/AVP is used, or

RTP/AVPF with T_rr_interval = 0, this is a non-issue, as the timeout period will be 25s, and differences between configured RTCP bandwidth can only cause premature timeouts when the reporting intervals are greater than 5s and differ by a factor of five. To limit the scope for such problematic misconfiguration, we propose an update to the RTP/AVPF timeout rules in Section 6.1.4.

6.1.3. Interoperability Between RTP/AVP and RTP/AVPF

If endpoints implementing the RTP/AVP and RTP/AVPF profiles (or their secure variants) are combined within a single RTP session, and the RTP/AVPF endpoints use a non-zero T_rr_interval that is significantly below 5 seconds, there is a risk that the RTP/AVPF endpoints will prematurely timeout the SSRCs of the RTP/AVP endpoints, due to their different RTCP timeout rules. Conversely, if the RTP/AVPF endpoints use a T_rr_interval that is significant larger than 5 seconds, there is a risk that the RTP/AVP endpoints will timeout the SSRCs of the RTP/AVPF endpoints.

Mixing endpoints using two different RTP profiles within a single RTP session is NOT RECOMMENDED. However, if mixed RTP profiles are used, and the RTP/AVPF endpoints are not updated to follow Section 6.1.4 of this memo, then the RTP/AVPF session SHOULD be configured to use T_rr_interval = 4 seconds to avoid premature timeouts.

The choice of T_rr_interval = 4 seconds for interoperability might appear strange. Intuitively, this value ought to be 5 seconds, to make both the RTP/AVP and RTP/AVPF use the same timeout period. However, the behaviour outlined in Section 6.1.1 shows that actual RTP/AVPF reporting intervals can be longer than expected. Setting T_rr_interval = 4 seconds gives actual RTCP intervals near to those expected by RTP/AVP, ensuring interoperability.

6.1.4. Updated SSRC Timeout Rules

To ensure interoperability and avoid premature timeouts, all SSRCs in an RTP session MUST use the same timeout behaviour. However, previous specification are inconsistent in this regard. To avoid interoperability issues, this memo updates the timeout rules as follows:

- o For the RTP/AVP, RTP/SAVP, RTP/AVPF, and RTP/SAVPF profiles, the timeout interval SHALL be calculated using a multiplier of five times the deterministic RTCP reporting interval. That is, the timeout interval SHALL be 5*Td.

- o For the RTP/AVP, RTP/SAVP, RTP/AVPF, and RTP/SAVPF profiles, calculation of Td, for the purpose of calculating the participant timeout only, SHALL be done using a Tmin value of 5 seconds and not the reduced minimal interval, even if the reduced minimum interval is used to calculate RTCP packet transmission intervals.

This changes the behaviour for the RTP/AVPF or RTP/SAVPF profiles when T_rr_interval != 0, a behaviour defined in Section 3.5.4 of RFC 4585, i.e. Tmin in the Td calculation is the T_rr_interval.

6.2. Tuning RTCP transmissions

This sub-section discusses what tuning can be done to reduce the downsides of the shared RTCP packet intervals. First, it is considered what possibilities exist for the RTP/AVP [RFC3551] profile, then what additional tools are provided by RTP/AVPF [RFC4585].

6.2.1. RTP/AVP and RTP/SAVP

When using the RTP/AVP or RTP/SAVP profiles, the options for tuning the RTCP reporting intervals are limited to the RTCP sender and receiver bandwidth, and whether the minimum RTCP interval is scaled according to the bandwidth. As the scheduling algorithm includes both randomisation and reconsideration, one cannot simply calculate the expected average transmission interval using the formula for Td given in Section 6.3.1 of [RFC3550]. However, by considering the inputs to that expression, and the randomisation and reconsideration rules, we can begin to understand the behaviour of the RTCP transmission interval.

Let’s start with some basic observations:

a. Unless the scaled minimum RTCP interval is used, then Td prior to randomization and reconsideration can never be less than Tmin. The default value of Tmin is 5 seconds.

b. If the scaled minimum RTCP interval is used, Td can become as low as 360 divided by RTP Session bandwidth in kilobits per second. In SDP the RTP session bandwidth is signalled using a "b=AS" line. An RTP Session bandwidth of 72kbps results in Tmin being 5 seconds. An RTP session bandwidth of 360kbps of course gives a Tmin of 1 second, and to achieve a Tmin equal to once every frame for a 25 frame-per-second video stream requires an RTP session bandwidth of 9Mbps. Use of the RTP/AVPF or RTP/SAVPF profile allows more frequent RTCP reports for the same bandwidth, as discussed below.

c. The value of Td scales with the number of SSRCs and the average size of the RTCP reports, to keep the overall RTCP bandwidth constant.

d. The actual transmission interval for a Td value is in the range [0.5*Td/1.21828,1.5*Td/1.21828], and the distribution is skewed, due to reconsideration, with the majority of the probability mass being above Td. This means, for example, that for Td = 5s, the actual transmission interval will be distributed in the range [2.052s, 6.156s], and tending towards the upper half of the interval. Note that Tmin parameter limits the value of Td before randomisation and reconsideration are applied, so the actual transmission interval will cover a range extending below Tmin.

Given the above, we can calculate the number of SSRCs, n, that an RTP session with 5% of the session bandwidth assigned to RTCP can support while maintaining Td equal to Tmin. This will tell us how many media streams we can report on, keeping the RTCP overhead within acceptable bounds. We make two assumptions that simplify the calculation: that all SSRCs are senders, and that they all send compound RTCP packets comprising an SR packet with n-1 report blocks, followed by an SDES packet containing a 16 octet CNAME value [RFC7022] (such RTCP packets will vary in size between 54 and 798 octets depending on n, up to the maximum of 31 report blocks that can be included in an SR packet). If we put this packet size, and a 5% RTCP bandwidth fraction into the RTCP interval calculation in Section 6.3.1 of [RFC3550], and calculate the value of n needed to give Td = Tmin for the scaled minimum interval, we find n=9 SSRCs can be supported (irrespective of the interval, due to the way the reporting interval scales with the session bandwidth). We see that to support more SSRCs, we need to increase the RTCP bandwidth fraction from 5%; changing the session bandwidth does not help due to the limit of Tmin.

To conclude, with RTP/AVP and RTP/SAVP the key limitation for small unicast sessions is going to be the Tmin value. Thus the RTP session bandwidth configured in RTCP has to be sufficiently high to reach the reporting goals the application has following the rules for the scaled minimal RTCP interval.

6.2.2. RTP/AVPF and RTP/SAVPF

When using RTP/AVPF or RTP/SAVPF, we have a powerful additional tool for tuning RTCP transmissions: the T_rr_interval parameter. Use of this parameter allows short RTCP reporting intervals; alternatively it gives the ability to sent frequent RTCP feedback without sending frequent regular RTCP reports.

The use of the RTP/AVPF or RTP/SAVPF profile with T_rr_interval set to a value greater than zero but smaller than Tmin allows more frequent RTCP feedback than the RTP/AVP or RTP/SAVP profiles, for a given RTCP bandwidth. This happens because Tmin is set to zero after the transmission of the initial RTCP report, causing the reporting interval for later packet to be determined by the usual RTCP bandwidth-based calculation, with Tmin=0, and the T_rr_interval. This has the effect that we are no longer restricted by the minimal interval (whether the default 5 second minimum, or the reduced minimum interval). Rather, the RTCP bandwidth and the T_rr_interval are the governing factors, allowing faster feedback. Applications that care about rapid regular RTCP feedback ought to consider using the RTP/AVPF or RTP/SAVPF profile, even if they don't use the feedback features of that profile.

The use of the RTP/AVPF or RTP/SAVPF profile allows RTCP feedback packets to be sent frequently, without also requiring regular RTCP reports to be sent frequently, since T_rr_interval limits the rate at which regular RTCP packets can be sent, while still permitting RTCP feedback packets to be sent. Applications that can use feedback packets for some media streams, e.g., video streams, but don't want frequent regular reporting for other media streams, can configure the T_rr_interval to a value so that the regular reporting for both audio and video is at a level that is considered acceptable for the audio. They could then use feedback packets, which will include RTCP SR/RR packets unless reduced size RTCP feedback packets [RFC5506] are used, for the video reporting. This allows the available RTCP bandwidth to be devoted on the feedback that provides the most utility for the application.

Using T_rr_interval still requires one to determine suitable values for the RTCP bandwidth value. Indeed, it might make this choice even more important, as this is more likely to affect the RTCP behaviour and performance than when using the RTP/AVP or RTP/SAVP profile, as there are fewer limitations affecting the RTCP transmission.

When T_rr_interval is non-zero, there are configurations that need to be avoided. If the RTCP bandwidth chosen is such that the Td value is smaller than, but close to, T_rr_interval, then the actual regular RTCP packet transmission interval can become very large, as discussed in Section 6.1.1. Therefore, for configuration where one intends to have Td smaller than T_rr_interval, then Td is RECOMMENDED to be targeted at values less than 1/4th of T_rr_interval which results in that the range becomes [0.5*T_rr_interval, 1.81*T_rr_interval].

With the RTP/AVPF or RTP/SAVPF profiles, using T_rr_interval = 0 has utility, and results in a behaviour where the RTCP transmission is only limited by the bandwidth, i.e., no Tmin limitations at all.

This allows more frequent regular RTCP reporting than can be achieved using the RTP/AVP profile. Many configurations of RTCP will not consume all the bandwidth that they have been configured to use, but this configuration will consume what it has been given. Note that the same behaviour will be achieved as long as T_rr_interval is smaller than 1/3 of Td as that prevents T_rr_interval from affecting the transmission.

There exists no method for using different regular RTCP reporting intervals depending on the media type or individual media stream, other than using a separate RTP session for each type or stream.

7. Security Considerations

When using the secure RTP protocol (RTP/SAVP) [RFC3711], or the secure variant of the feedback profile (RTP/SAVPF) [RFC5124], the cryptographic context of a compound secure RTCP packet is the SSRC of the sender of the first RTCP (sub-)packet. This could matter in some cases, especially for keying mechanisms such as Mikey [RFC3830] which allow use of per-SSRC keying.

Otherwise, the standard security considerations of RTP apply; sending multiple media streams from a single endpoint in a single RTP session does not appear to have different security consequences than sending the same number of media streams spread across different RTP sessions.

8. IANA Considerations

No IANA actions are needed.

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.

[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.

[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.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

9.2. Informative References

[I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-07 (work in progress), March 2015.

[I-D.ietf-avtcore-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-05 (work in progress), February 2015.

[I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-06 (work in progress), March 2015.

[I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-21 (work in progress), March 2015.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-18 (work in progress), March 2015.

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

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

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

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

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org

AVTCORE Working Group J. Mattsson, Ed.
Internet-Draft Ericsson
Intended status: Standards Track D. McGrew
Expires: April 23, 2015 D. Wing
F. Andreasen
Cisco
October 20, 2014

Encrypted Key Transport for Secure RTP
draft-ietf-avtcore-srtp-ekt-03

Abstract

Encrypted Key Transport (EKT) is an extension to Secure Real-time Transport Protocol (SRTP) that provides for the secure transport of SRTP master keys, Rollover Counters, and other information. This facility enables SRTP to work for decentralized conferences with minimal control.

This note defines EKT, and also describes how to use it with SDP Security Descriptions, DTLS-SRTP, and MIKEY. With EKT, these other key management protocols provide an EKT key to everyone in a session, and EKT coordinates the SRTP keys within the session.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RTP is designed to allow decentralized groups with minimal control to establish sessions, such as for multimedia conferences. Unfortunately, Secure RTP (SRTP [RFC3711]) cannot be used in many minimal-control scenarios, because it requires that SSRC values and other data be coordinated among all of the participants in a session. For example, if a participant joins a session that is already in progress, that participant needs to be told the SRTP keys (and SSRC, ROC and other details) of the other SRTP sources.

The inability of SRTP to work in the absence of central control was well understood during the design of the protocol; the omission was considered less important than optimizations such as bandwidth conservation. Additionally, in many situations SRTP is used in conjunction with a signaling system that can provide most of the central control needed by SRTP. However, there are several cases in which conventional signaling systems cannot easily provide all of the coordination required. It is also desirable to eliminate the layer violations that occur when signaling systems coordinate certain SRTP parameters, such as SSRC values and ROCs.

This document defines Encrypted Key Transport (EKT) for SRTP, an extension to SRTP that fits within the SRTP framework and reduces the amount of external signaling control that is needed in an SRTP session. EKT securely distributes the SRTP master key and other information for each SRTP source (SSRC), using SRTCP or SRTP to transport that information. With this method, SRTP entities are free to choose SSRC values as they see fit, and to start up new SRTP sources (SSRC) with new SRTP master keys (see Section 2.2) within a session without coordinating with other entities via external signaling or other external means. This fact allows to reinstate the RTP collision detection and repair mechanism, which is nullified by the current SRTP specification because of the need to control SSRC values closely. An SRTP endpoint using EKT can generate new keys whenever an existing SRTP master key has been overused, or start up a new SRTP source (SSRC) to replace an old SRTP source that has reached the packet-count limit. However, EKT does not allow SRTP's ROC to rollover; that requires re-keying outside of EKT (e.g., using MIKEY or DTLS-SRTP). EKT also solves the problem in which the burst loss of the N initial SRTP packets can confuse an SRTP receiver, when the initial RTP sequence number is greater than or equal to 2^16 - N. These features can simplify many architectures that implement SRTP.

EKT provides a way for an SRTP session participant, either a sender or receiver, to securely transport its SRTP master key and current SRTP rollover counter to the other participants in the session. This data, possibly in conjunction with additional data provided by an external signaling protocol, furnishes the information needed by the receiver to instantiate an SRTP/SRTCP receiver context.

EKT does not control the manner in which the SSRC is generated; it is only concerned with their secure transport. Those values may be generated on demand by the SRTP endpoint, or may be dictated by an external mechanism such as a signaling agent or a secure group controller.

EKT is not intended to replace external key establishment mechanisms such as SDP Security Descriptions [RFC4568], DTLS-SRTP [RFC5764], or MIKEY [RFC3830][RFC4563]. Instead, it is used in conjunction with those methods, and it relieves them of the burden of tightly coordinating every SRTP source (SSRC) among every SRTP participant.

1.1. History

[[RFC Editor Note: please remove this section prior to publication as an RFC.]]

A substantial change occurred between the EKT documents draft-ietf-avt-srtp-ekt-03 and draft-ietf-avtcore-srtp-ekt-00. The change makes it possible for the EKT data to be removed from a packet without affecting the ability of the receiver to correctly process the data that is present in that packet. This capability facilitates interoperability between SRTP implementations with different SRTP key management methods. The changes also greatly simplify the EKT processing rules, and makes the EKT data that must be carried in SRTP and/or SRTCP packets somewhat larger.

In draft-ietf-avtcore-srtp-ekt-02, SRTP master keys have to be always generated randomly and not re-used, MKI is no longer allowed with EKT (as MKI duplicates some of EKT's functions), and text clarifies that EKT must be negotiated during call setup. Some text was consolidated and re-written, notably Section 2.6 ("Timing and Reliability"). Support for re-directing the DTLS-SRTP handshake to another host was removed, as it needed NAT traversal support.

In draft-ietf-avtcore-srtp-ekt-03, the SRTCP compound packet problem is discussed. Updates and clarifications were made to the SDESC and MIKEY sections.

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].

2. Encrypted Key Transport

In EKT, an SRTP master key is encrypted with a key encrypting key and the resulting ciphertext is transported in selected SRTCP packets or in selected SRTP packets. The key encrypting key is called an EKT key. A single such key suffices for a single SRTP session, regardless of the number of participants in that session. However, there can be multiple EKT keys used within a particular session.

EKT defines a new method of providing SRTP master keys to an endpoint. In order to convey the ciphertext of the SRTP master key, and other additional information, an additional EKT field is added to SRTP or SRTCP packets. When added to SRTCP, the EKT field appears at the end of the packet, after the authentication tag, if that tag is present, or after the SRTCP index otherwise. When added to SRTP, The EKT field appears at the end of the SRTP packet, after the authentication tag (if that tag is present), or after the ciphertext of the encrypted portion of the packet otherwise.

EKT MUST NOT be used in conjunction with SRTP's MKI (Master Key Identifier) or with SRTP's <From, To> [RFC3711], as those SRTP features duplicate some of the functions of EKT.

2.1. EKT Field Formats

The EKT Field uses one of the two formats defined below. These two formats can always be unambiguously distinguished on receipt by examining the final bit of the EKT Field, which is also the final bit of the SRTP or SRTCP packet. The first format is the Full EKT Field (or Full_EKT_Field), and the second is the Short EKT Field (or Short_EKT_Field). The formats are defined as

```
EKT_Plaintext = SRTP_Master_Key || SSRC || ROC || ISN

EKT_Ciphertext = EKT_Encrypt(EKT_Key, EKT_Plaintext)

Full_EKT_Field = EKT_Ciphertext || SPI || '1'

Short_EKT_Field = Reserved || '0'
```

Figure 1: EKT data formats

Here || denotes concatenation, and '1' and '0' denote single one and zero bits, respectively. These fields and data elements are defined as follows:

EKT_Plaintext: The data that is input to the EKT encryption operation. This data never appears on the wire, and is used only in computations internal to EKT.

EKT_Ciphertext: The data that is output from the EKT encryption operation, described in Section 2.3. This field is included in SRTP and SRTCP packets when EKT is in use. The length of this field is variable, and is equal to the ciphertext size N defined in Section 2.3. Note that the length of the field is inferable from the SPI field, since the particular EKT cipher used by the sender of a packet can be inferred from that field.

SRTP_Master_Key: On the sender side, the SRTP Master Key associated with the indicated SSRC. The length of this field depends on the cipher suite negotiated during call setup for SRTP or SRTCP.

SSRC: On the sender side, this field is the SSRC for this SRTP source. The length of this field is fixed at 32 bits.

Rollover Counter (ROC): On the sender side, this field is set to the current value of the SRTP rollover counter in the SRTP context associated with the SSRC in the SRTP or SRTCP packet. The length of this field is fixed at 32 bits.

Initial Sequence Number (ISN): If this field is nonzero, it indicates the RTP sequence number of the initial RTP packet that is protected using the SRTP master key conveyed (in encrypted form) by the EKT Ciphertext field of this packet. When this field is present in an RTCP packet it indicates the RTP sequence number of the first RTP packet encrypted by this master key. If the ISN field is zero, it indicates that the initial RTP/RTCP packet protected using the SRTP master key conveyed in this packet preceded, or was concurrent with, the last roll-over of the RTP sequence number, and thus should be used as the current master key for processing this packet. The length of this field is fixed at 16 bits.

Security Parameter Index (SPI): This field is included in SRTP and SRTCP packets when EKT is in use. It indicates the appropriate EKT key and other parameters for the receiver to use when processing the packet. It is an "index" into a table of possibilities (which are established via signaling or some other out-of-band means), much like the IPsec Security Parameter Index [RFC4301]. The length of this field is fixed at 15 bits. The parameters identified by this field are:

* The EKT key used to process the packet.

* The EKT cipher used to process the packet.

* The Secure RTP parameters associated with the SRTP Master Key carried by the packet and the SSRC value in the packet. Section 8.2. of [RFC3711] summarizes the parameters defined by that specification.

* The Master Salt associated with the Master Key. (This value is part of the parameters mentioned above, but we call it out for emphasis.) The Master Salt is communicated separately, via signaling, typically along with the EKT key.

Together, these data elements are called an EKT parameter set. Within each SRTP session, each distinct EKT parameter set that may be used MUST be associated with a distinct SPI value, to avoid ambiguity.

Reserved: The length of this field is 7 bits. MUST be all zeros on transmission, and MUST be ignored on reception.

The Full_EKT_Field and Short_EKT_Field formats are shown in Figure 2 and Figure 3, respectively. These figures show the on-the-wire data. The Ciphertext field holds encrypted data, and thus has no apparent inner structure.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                        EKT Ciphertext                         :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Security Parameter Index  |1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Full EKT Field format

```
 0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+
|   Reserved    |0|
+-+-+-+-+-+-+-+-+-+
```

Figure 3: Short EKT Field format

2.2. Packet Processing and State Machine

At any given time, each SRTP/SRTCP source (SSRC) has associated with it a single EKT parameter set. This parameter set is used to process all outbound packets, and is called the outbound parameter set. There may be other EKT parameter sets that are used by other SRTP/SRTCP sources in the same session, including other SRTP/SRTCP sources on the same endpoint (e.g., one endpoint with voice and video might have two EKT parameter sets, or there might be multiple video sources on an endpoint each with their own EKT parameter set). All of these EKT parameter sets SHOULD be stored by all of the participants in an SRTP session, for use in processing inbound SRTP and SRTCP traffic.

All SRTP master keys MUST NOT be re-used, MUST be randomly generated according to [RFC4086], and MUST NOT be equal to or derived from other SRTP master keys.

2.2.1. Outbound Processing

See Section 2.6 which describes when to send an EKT packet and describes if a Full EKT Field or Short EKT Field is sent.

When an SRTP or SRTCP packet is to be sent, the EKT field for that packet is created as follows, or uses an equivalent set of steps. The creation of the EKT field MUST precede the normal SRTP or SRTCP packet processing. The ROC used in EKT processing MUST be the same as the one used in the SRTP processing.

If the Short format is used, an all-zero octet is appended to the packet. Otherwise, processing continues as follows.

The Rollover Counter field in the packet is set to the current value of the SRTP rollover counter (represented as an unsigned integer in network byte order).

The Initial Sequence Number field is set to zero, if the initial RTP packet protected using the current SRTP master key for this source preceded, or was concurrent with, the last roll-over of the RTP sequence number. Otherwise, that field is set to the value of the RTP sequence number of the initial RTP packet that was or will be protected by that key. See "rekey" in Section 2.6. The rekeying event MUST NOT change the value of ROC (otherwise, the current value of the ROC would not be known to late joiners of existing sessions). This means rekeying near the end of sequence number space (e.g., 100 packets before sequence number 65535) is not possible because ROC needs to roll over.

The Security Parameter Index field is set to the value of the Security Parameter Index that is associated with the outbound parameter set.

The EKT_Plaintext field is computed from the SRTP Master Key, SSRC, ROC, and ISN fields, as shown in Figure 1.

The EKT_Ciphertext field is set to the ciphertext created by encrypting the EKT_Plaintext with the EKT cipher, using the EKT Key as the encryption key. The encryption process is detailed in Section 2.3. Implementations MAY cache the value of this field to avoid recomputing it for each packet that is sent.

Implementation note: Because of the format of the Full EKT Field, a packet containing the Full EKT Field MUST be sent when the ROC changes (i.e., every 2^16 packets).

2.2.2. Inbound Processing

When an SRTP or SRTCP packet containing a Full EKT Field or Short EKT Field is received, it is processed as follows or using an equivalent set of steps. Inbound EKT processing MUST take place prior to the usual SRTP or SRTCP processing. Implementation note: the receiver may want to have a sliding window to retain old master keys for some brief period of time, so that out of order packets can be processed. The following steps show processing as packets are received in order.

1. The final bit is checked to determine which EKT format is in use. If the packet contains a Short EKT Field then the Short EKT Field

is removed and normal SRTP or SRTCP processing is applied. If the packet contains a Full EKT Field, then processing continues as described below.

2. The Security Parameter Index (SPI) field is checked to determine which EKT parameter set should be used when processing the packet. If multiple parameter sets have been defined for the SRTP session, then the one that is associated with the value of the SPI field in the packet is used. This parameter set is called the matching parameter set below. If there is no matching SPI, then the verification function MUST return an indication of authentication failure, and the steps described below are not performed.

3. The EKT_Ciphertext is decrypted using the EKT_Key and EKT_Cipher in the matching parameter set, as described in Section 2.3. If the EKT decryption operation returns an authentication failure, then the packet processing halts with an indication of failure. Otherwise, the resulting EKT_Plaintext is parsed as described in Figure 1, to recover the SRTP Master Key, SSRC, ROC, and ISN fields.

4. The SSRC field output from the decryption operation is compared to the SSRC field from the SRTP header if EKT was received over SRTP, or from the SRTCP header if EKT was received over SRTCP. If the values of the two fields do not match, then packet processing halts with an indication of failure. Otherwise, it continues as follows.

5. If an SRTP context associated with the SSRC in the previous step already exists and the ROC from the EKT_Plaintext is less than the ROC in the SRTP context, then EKT processing halts and the packet is processed as an out-of-order packet (if within the implementation's sliding window) or dropped (as it is a replay). Otherwise, the ROC in the SRTP context is set to the value of the ROC from the EKT_Plaintext, and the SRTP Master Key from the EKT_Plaintext is accepted as the SRTP master key corresponding to the SSRC indicated in the EKT_Plaintext, beginning at the sequence number indicated by the ISN (see next step).

6. If the ISN from the EKT_Plaintext is less than the RTP sequence number of an authenticated received SRTP packet, then EKT processing halts (as this is a replay). If the Initial Sequence Number field is nonzero, then the initial sequence number for the SRTP master key is set to the packet index created by appending that field to the current rollover counter and treating the result as a 48-bit unsigned integer. The initial sequence number for the master key is equivalent to the "From" value of the

<From, To> pair of indices (Section 8.1.1 of [RFC3711]) that can be associated with a master key.

7. The newly accepted SRTP master key, the SRTP parameters from the matching parameter set, and the SSRC from the packet are stored in the crypto context associated with the SRTP source (SSRC). The SRTP Key Derivation algorithm is run in order to compute the SRTP encryption and authentication keys, and those keys are stored for use in SRTP processing of inbound packets. The Key Derivation algorithm takes as input the newly accepted SRTP master key, along with the Master Salt from the matching parameter set.

8. At this point, EKT processing has successfully completed, and the normal SRTP or SRTCP processing takes place.

   Implementation note: the value of the EKT Ciphertext field is identical in successive packets protected by the same EKT parameter set and the same SRTP master key, ROC, and ISN. This ciphertext value MAY be cached by an SRTP receiver to minimize computational effort by noting when the SRTP master key is unchanged and avoiding repeating Steps 2 through 6.

2.3. Ciphers

EKT uses an authenticated cipher to encrypt the EKT Plaintext, which is comprised of the SRTP master keys, SSRC, ROC, and ISN. We first specify the interface to the cipher, in order to abstract the interface away from the details of that function. We then define the cipher that is used in EKT by default. The default cipher described in Section 2.3.1 MUST be implemented, but another cipher that conforms to this interface MAY be used, in which case its use MUST be coordinated by external means (e.g., key management).

The master salt length for the offered cipher suites MUST be the same. In practice the easiest way to achieve this is by offering the same crypto suite.

An EKT cipher consists of an encryption function and a decryption function. The encryption function E(K, P) takes the following inputs:

- a secret key K with a length of L bytes, and

- a plaintext value P with a length of M bytes.

The encryption function returns a ciphertext value C whose length is N bytes, where N is at least M. The decryption function D(K, C) takes the following inputs:

- a secret key K with a length of L bytes, and
- a ciphertext value C with a length of N bytes.

The decryption function returns a plaintext value P that is M bytes long, or returns an indication that the decryption operation failed because the ciphertext was invalid (i.e. it was not generated by the encryption of plaintext with the key K).

These functions have the property that D(K, E(K, P)) = P for all values of K and P. Each cipher also has a limit T on the number of times that it can be used with any fixed key value. For each key, the encryption function MUST NOT be invoked on more than T distinct values of P, and the decryption function MUST NOT be invoked on more than T distinct values of C.

The length of the EKT Plaintext is ten bytes, plus the length of the SRTP Master Key.

Security requirements for EKT ciphers are discussed in Section 8.

2.3.1. The Default Cipher

The default EKT Cipher is the Advanced Encryption Standard (AES) [FIPS197] Key Wrap with Padding [RFC5649] algorithm. It requires a plaintext length M that is at least one octet, and it returns a ciphertext with a length of N = M + 8 octets. It can be used with key sizes of L = 16, 24, and 32, and its use with those key sizes is indicated as AESKW_128, AESKW_192, and AESKW_256, respectively. The key size determines the length of the AES key used by the Key Wrap algorithm. With this cipher, T=2^48.

SRTP transform	EKT transform	length of EKT plaintext	length of EKT ciphertext	length of Full EKT Field
AES-128	AESKW_128 (m)	26	40	42
AES-192	AESKW_192	34	48	50
AES-256	AESKW_256	42	56	58
F8-128	AESKW_128	26	40	42
SEED-128	AESKW_128	26	40	42

Figure 4: AESKW Table

The mandatory to implement transform is AESKW_128, indicated by (m).

As AES-128 is the mandatory to implement transform in SRTP [RFC3711], AESKW_128 MUST be implemented for EKT.

For all the SRTP transforms listed in the table, the corresponding EKT transform MUST be used, unless a stronger EKT transform is negotiated by key management.

2.3.2. Other EKT Ciphers

Other specifications may extend this one by defining other EKT ciphers per Section 9. This section defines how those ciphers interact with this specification.

An EKT cipher determines how the EKT Ciphertext field is written, and how it is processed when it is read. This field is opaque to the other aspects of EKT processing. EKT ciphers are free to use this field in any way, but they SHOULD NOT use other EKT or SRTP fields as an input. The values of the parameters L, M, N, and T MUST be defined by each EKT cipher, and those values MUST be inferable from the EKT parameter set.

2.4. Synchronizing Operation

A participant in a session MAY opt to use a particular EKT parameter set to protect outbound packets after it accepts that EKT parameter set for protecting inbound traffic. In this case, the fact that one participant has changed to using a new EKT key for outbound traffic can trigger other participants to switch to using the same key.

If a source has its EKT key changed by key management, it MUST also change its SRTP master key, which will cause it to send out a new Full EKT Field. This ensures that if key management thought the EKT key needs changing (due to a participant leaving or joining) and communicated that in key management to a source, the source will also change its SRTP master key, so that traffic can be decrypted only by those who know the current EKT key.

The use of EKT MUST be negotiated during key management or call setup (e.g., using DTLS-SRTP, Security Descriptions, MIKEY, or similar).

2.5. Transport

EKT SHOULD be used over SRTP, and MAY be used over SRTCP. SRTP is preferred because it shares fate with transmitted media, because SRTP rekeying can occur without concern for RTCP transmission limits, and to avoid SRTCP compound packets with RTP translators and mixers.

This specification requires the EKT SSRC match the SSRC in the RTCP header, but Section 6.1 of [RFC3550] encourages creating SRTCP compound packets:

> It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7).

These compound SRTCP packets might have an SSRC that does not match the EKT SSRC. To reduce the occasion of this occuring, EKT-aware RTP mixers and translators which are generating SRTCP compound packets SHOULD attempt to place an SRTCP payload containing an EKT tag at the front of the compound packet (so that the EKT receiver will process it), and MAY be even more robust and implement more sophisticated algorithms to ensure all EKT tags from different senders are sent at the front of the compound packet. However, no robust algorithm exists which ensures robust EKT delivery in conjunction with SRTCP compound packets. This impact to RTP translators and mixers, and the inability to realibly determine an RTP translator or mixer might be involved in an RTP session, provides further incentive to send EKT over RTP.

The packet processing, state machine, and Authentication Tag format for EKT over SRTP are nearly identical to that for EKT over SRTCP. Differences are highlighted in Section 2.2.1 and Section 2.2.2.

The Full EKT Field is appended to the SRTP or SRTCP payload and is 42, 50, or 58 octets long for AES-128, AES-192, or AES-256, respectively. This length impacts the maximum payload size of the SRTP (or SRTCP) packet itself. To remain below the network path MTU, senders SHOULD constrain the SRTP (or SRTCP) payload size by this length of the Full EKT Field.

EKT can be transported over SRTCP, but some of the information that it conveys is used for SRTP processing; some elements of the EKT parameter set apply to both SRTP and SRTCP. Furthermore, SRTCP packets can be lost and both SRTP and SRTCP packets may be delivered out of order. This can lead to various race conditions if EKT is transported over SRTCP but not SRTP, which we review below.

The ROC signaled via EKT over SRTCP may be off by one when it is received by the other party(ies) in the session. In order to deal with this, receivers should simply follow the SRTP packet index estimation procedures defined in Section 3.3.1 [RFC3711].

2.6. Timing and Reliability Consideration

A system using EKT has the SRTP master keys distributed with EKT, rather than with call signaling. A receiver can immediately decrypt an SRTP (or SRTCP packet) using that new key, provided the SRTP packet (or SRTCP packet) also contains a Full EKT Field.

This section describes how to reliably and expediently deliver new SRTP master keys to receivers.

There are three cases to consider. The first case is a new sender joining a session which needs to communicate its SRTP master key to all the receivers. The second case is a sender changing its SRTP master key which needs to be communicated to all the receivers. The third case is a new receiver joining a session already in progress which needs to know the sender's SRTP master key.

New sender: A new sender SHOULD send a packet containing the Full EKT Field as soon as possible, always before or coincident with sending its initial SRTP packet. To accommodate packet loss, it is RECOMMENDED that three consecutive packets contain the Full EKT Field be transmitted. Inclusion of that Full EKT Field can be stopped early if the sender determines all receivers have received the new SRTP master key by receipt of an SRTCP receiver report or explicit ACK for a sequence number with the new key.

Rekey: By sending EKT over SRTP, the rekeying event shares fate with the SRTP packets protected with that new SRTP master key. To avoid sending large SRTP packets (such as video key frames) with the Full EKT Field, it can be advantageous to send smaller SRTP packets with the Full EKT Field with the Initial Sequence Number prior to the actual rekey event, but this does eliminate the benefits of fate-sharing EKT with the SRTP packets with the new SRTP master key, which increases the chance a new receiver won't have seen the new SRTP master key.

New receiver: When a new receiver joins a session it does not need to communicate its sending SRTP master key (because it is a receiver). When a new receiver joins a session the sender is generally unaware of the receiver joining the session. Thus, senders SHOULD periodically transmit the Full EKT Field. That interval depends on how frequently new receivers join the session, the acceptable delay before those receivers can start processing SRTP packets, and the acceptable overhead of sending the Full EKT Field. The RECOMMENDED frequency is the same as the key frame frequency if sending video or every 5 seconds. When joining a session it is likely that SRTP or SRTCP packets might be received before a packet containing the Full EKT Field is received. Thus, to avoid doubling the authentication effort, an implementation joining an EKT session SHOULD buffer received SRTP and SRTCP packets until it receives the Full EKT Field packet and use the information in that packet to authenticate and decrypt the received SRTP/SRTCP packets.

3. Use of EKT with SDP Security Descriptions

The SDP Security Descriptions (SDESC) [RFC4568] specification defines a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol with the "crypto" attribute and the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDESC also defines how to use that framework specifically to negotiate security parameters for Secure RTP. Below, we first provide a brief recap of the crypto attribute when used for SRTP and we then explain how it is complementary to EKT. In the rest of this Section, we provide extensions to the crypto attribute and associated offer/answer procedures to define its use with EKT.

3.1. SDP Security Descriptions Recap

The SRTP crypto attribute defined for SDESC contains a tag followed by three types of parameters (refer to [RFC4568] for details):

- o Crypto-suite. Identifies the encryption and authentication transform.

- o Key parameters. SRTP keying material and parameters.

- o Session parameters. Additional (optional) SRTP parameters such as Key Derivation Rate, Forward Error Correction Order, use of unencrypted SRTP, and other parameters defined by SDESC.

The crypto attributes in the example SDP in Figure 5 illustrate these parameters.

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
   inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20
   FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
   inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjiiKt|2^20
   FEC_ORDER=FEC_SRTP
```

Figure 5: SDP Security Descriptions example

For legibility the SDP shows line breaks that are not present on the wire.

The first crypto attribute has the tag "1" and uses the crypto-suite AES_CM_128_HMAC_SHA1_80. The "inline" parameter provides the SRTP master key and salt and the master key lifetime (number of packets). Finally, the FEC_ORDER session parameter indicates the order of Forward Error Correction used (FEC is applied before SRTP processing by the sender of the SRTP media).

The second crypto attribute has the tag "2", the crypto-suite F8_128_HMAC_SHA1_80, a SRTP master key, and its associated salt. Finally, the FEC_ORDER session parameter indicates the order of Forward Error Correction used.

3.2. Relationship between EKT and SDESC

SDP Security Descriptions [RFC4568] define a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol by use of the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDESC also defines how to use it specifically to negotiate security parameters for Secure RTP.

EKT and SDP Security Descriptions are complementary. SDP Security Descriptions can negotiate several of the SRTP security parameters (e.g., cipher and use of Master Key Identifier) as well as SRTP master keys. SDESC, however, does not negotiate SSRCs and their associated Rollover Counter (ROC). Instead, SDESC relies on a so-called "late binding", where a newly observed SSRC will have its crypto context initialized to a ROC value of zero. Clearly, this does not work for participants joining an SRTP session that has been established for a while and hence has a non-zero ROC. It is impossible to use SDESC to join an SRTP session that is already in progress. In this case, EKT on the endpoint running SDESC can provide the additional signaling necessary to communicate the ROC (Section 6.4.1 of [RFC4568]). The use of EKT solves this problem by communicating the ROC associated with the SSRC in the media plane.

SDP Security Descriptions negotiates different SRTP master keys in the send and receive direction. The offer contains the master key used by the offerer to send media, and the answer contains the master key used by the answerer to send media. Consequently, if media is received by the offerer prior to the answer being received, the offerer does not know the master key being used. Use of SDP security preconditions can solve this problem, however it requires an additional round-trip as well as a more complicated state machine. EKT solves this problem by simply sending the master key used in the media plane thereby avoiding the need for security preconditions.

If multiple crypto-suites were offered, the offerer also will not know which of the crypto-suites offered was selected until the answer is received. EKT solves this problem by using a correlator, the Security Parameter Index (SPI), which uniquely identifies each crypto attribute in the offer.

One of the primary call signaling protocols using offer/answer is the Session Initiation Protocol (SIP) [RFC3261]. SIP uses the INVITE message to initiate a media session and typically includes an offer SDP in the INVITE. An INVITE may be "forked" to multiple recipients which potentially can lead to multiple answers being received. SDESC, however, does not properly support this scenario, mainly because SDP and RTP/RTCP does not contain sufficient information to allow for correlation of an incoming RTP/RTCP packet with a particular answer SDP. Note that extensions providing this correlation do exist (e.g., Interactive Connectivity Establishment (ICE)). SDESC addresses this point-to-multipoint problem by moving each answer to a separate RTP transport address thereby turning a point-to-multipoint scenario into multiple point-to-point scenarios. There are however significant disadvantages to doing so. As long as the crypto attribute in the answer does not contain any declarative parameters that differ from those in the offer, EKT solves this problem by use of the SPI correlator and communication of the answerer's SRTP master key in EKT.

As can be seen from the above, the combination of EKT and SDESC provides a better solution to SRTP negotiation for offer/answer than either of them alone. SDESC negotiates the various SRTP crypto parameters (which EKT does not), whereas EKT addresses some of the shortcomings of SDESC.

3.3. Overview of Combined EKT and SDESC Operation

We define a new session parameter to SDESC to communicate the EKT cipher, EKT key, and Security Parameter Index to the peer. The original SDESC parameters are used as defined in [RFC4568], however the procedures associated with the SRTP master key differ slightly, since both SDESC and EKT communicate an SRTP master key. In particular, the SRTP master key communicated via SDESC is used only if there is currently no crypto context established for the SSRC in question. This will be the case when an entity has received only the offer or answer, but has yet to receive a valid EKT packet from the peer. Once a valid EKT packet is received for the SSRC, the crypto context is initialized accordingly, and the SRTP master key will then be derived from the EKT packet. Subsequent offer/answer exchanges do not change this: The most recent SRTP master key negotiated via EKT will be used, or, if none is available for the SSRC in question, the most recent SRTP master key negotiated via offer/answer will be used. This is done to avoid race conditions between the offer/answer exchange and EKT, even though this breaks some offer/answer rules. Note that with the rules described in this paragraph, once a valid EKT packet has been received for a given SSRC, rekeying for that SSRC can only be done via EKT. The associated SRTP crypto parameters however can be changed via SDESC.

3.4. EKT Extensions to SDP Security Descriptions

In order to use EKT and SDESC in conjunction with each other, the new SDESC session parameter "EKT" is defined. It MUST NOT appear more than once in a given crypto attribute. In the Offer/Answer model, the EKT parameter is a negotiated parameter.The "EKT" session parameter consists of three parts (the formal grammar is provided in Section 3.9):

"EKT=" <EKT_Cipher> "|" <EKT_Key> "|" <EKT_SPI>

Below are details on each of these attributes.

EKT_Cipher: The (optional) EKT_Cipher field defines the EKT cipher used to encrypt the EKT key within SRTP and SRTCP packets. The default value is "AESKW_128" in accordance with Section 2.3.1. For the AES Key Wrap cipher, the values "AESKW_128", "AESKW_192", and "AESKW_256" are defined for values of L=16, 24, and 32 respectively.

EKT_Key: The (mandatory) EKT_Key field is the EKT key used to encrypt the SRTP Master Key within SRTP and SRTCP packets. The value is base64 encoded with "=" padding if padding is necessary (see Section 3.2 and 4 of [RFC4648]).

EKT_SPI: The (mandatory) EKT_SPI field is the Security Parameter Index. It is encoded as an ASCII string representing the hexadecimal value of the Security Parameter Index. The SPI identifies the *offer* crypto attribute (including the EKT Key and Cipher) being used for the associated SRTP session. A crypto attribute corresponds to an EKT Parameter Set and hence the SPI effectively identifies a particular EKT parameter set. Note that the scope of the SPI is the SRTP session, which may or may not be limited to the scope of the associated SIP dialog. In particular, if one of the participants in an SRTP session is an SRTP translator, the scope of the SRTP session is not limited to the scope of a single SIP dialog. However, if all of the participants in the session are endpoints or mixers, the scope of the SRTP session will correspond to a single SIP dialog.

3.5. Offer/Answer Considerations

In this section, we provide the offer/answer procedures associated with use of the new SDESC session parameter defined in Section 3.4. Since SDESC is defined only for unicast streams, we provide only offer/answer procedures for unicast streams here as well.

3.5.1. Generating the Initial Offer - Unicast Streams

When the initial offer is generated, the offerer MUST follow the steps defined in [RFC4568] Section 7.1.1 as well as the following steps.

[[Editor’s Note: following paragraph would benefit from rewording.]]

For each unicast media line using Security Descriptions and where use of EKT is desired, the offerer MUST include the EKT parameter in at least one "crypto" attribute (see [RFC4568]). The EKT paramater MUST contain the EKT_Key and EKT_SPI fields. The EKT_SPI field serves to identify the EKT parameter set used for a particular SRTP or SRTCP packet. Consequently, within a single media line, a given EKT_SPI value MUST NOT be used with multiple crypto attributes. Note that the EKT parameter set to use for the session is not yet established at this point; each offered crypto attribute contains a candidate EKT parameter set. Furthermore, if the media line refers to an existing SRTP session, then any SPI values used for EKT parameter sets in that session MUST NOT be remapped to any different EKT parameter sets. When an offer describes an SRTP session that is already in progress, the offer SHOULD use an EKT parameter set (including EKT_SPI and EKT_KEY) that is already in use.

As EKT is not defined for use with MKI, a "crypto" attribute containing the EKT parameter MUST NOT contain MKI.

Important Note: The scope of the offer/answer exchange is the SIP dialog(s) established as a result of the INVITE, however the scope of EKT is the direct SRTP session, i.e., all the participants that are able to receive SRTP and SRTCP packets directly. If an SRTP session spans multiple SIP dialogs, the EKT parameter sets MUST be synchronized between all the SIP dialogs where SRTP and SRTCP packets can be exchanged. In the case where the SIP entity operates as an RTP mixer (and hence re-originates SRTP and SRTCP packets with its own SSRC), this is not an issue, unless the mixer receives traffic from the various participants on the same destination IP address and port, in which case further coordination of SPI values and crypto parameters may be needed between the SIP dialogs (note that SIP forking with multiple early media senders is an example of this). However, if it operates as a transport translator (relay) then synchronized negotiation of the EKT parameter sets on *all* the involved SIP dialogs will be needed. This is non-trivial in a variety of use cases, and hence use of the combined SDES/EKT mechanism with RTP translators should be considered very carefully. It should be noted, that use of SRTP with RTP translators in general should be considered very carefully as well.

The session parameter "EKT" can either be included as an optional or mandatory parameter.

3.5.2. Generating the Initial Answer - Unicast Streams

When the initial answer is generated, the answerer MUST follow the steps defined in [RFC4568] Section 7.1.2 as well as the following steps.

For each unicast media line using SDESC, the answerer examines the associated crypto attribute(s) for the presence of the session parameter "EKT". If a mandatory EKT parameter is included with a "crypto" attribute, the answerer MUST support those parameters in order to accept that offered crypto attribute. If an optional EKT parameter is included instead, the answerer MAY accept the offered crypto attribute without using EKT. However, doing so will prevent the offerer from processing any packets received before the answer. If no EKT parameter are included with a crypto attribute, and that crypto attribute is accepted in the answer, EKT MUST NOT be used. If a given a crypto attribute includes a malformed EKT parameter, that crypto attribute MUST be considered invalid.

When EKT is used with SDESC, the offerer and answerer MUST use the same SRTP master salt. Thus, the SRTP key parameter(s) in the answer crypto attribute MUST use the same master salt as the one accepted from the offer.

When the answerer accepts the offered media line and EKT is being used, the crypto attribute included in the answer MUST include the same EKT parameter values as found in the accepted crypto attribute from the offerer (however, if the default EKT cipher is being used, it may be omitted). Furthermore, the EKT parameter included MUST be mandatory (i.e., no "-" prefix).

Acceptance of a crypto attribute with an EKT parameter leads to establishment of the EKT parameter set for the corresponding SRTP session. Consequently, the answerer MUST send packets in accordance with that particular EKT parameter set only. If the answerer wants to enable the offerer to process SRTP packets received by the offerer before it receives the answer, the answerer MUST NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value. Otherwise, the offerer's view of the EKT parameter set would differ from the answerer's until the answer is received. Similarly, unless the offerer and answerer has other means for correlating an answer with a particular SRTP session, the answer SHOULD NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value. If this recommendation is not followed and the offerer receives multiple answers (e.g., due to SIP forking), the offerer may not be able to process incoming media stream packets correctly.

3.5.3. Processing of the Initial Answer - Unicast Streams

When the offerer receives the answer, it MUST perform the steps in [RFC4568] Section 7.1.3 as well as the following steps for each SRTP media stream it offered with one or more crypto lines containing EKT parameters in it.

[[Editor's Note: following paragraph would benefit from rewording.]]

If the answer crypto line contains an EKT parameter, and the corresponding crypto line from the offer contained the same EKT values, use of EKT has been negotiated successfully and MUST be used for the media stream. When determining whether the values match, an optional and mandatory parameter MUST be considered equal.

Furthermore, if the default EKT cipher is being used, it MAY be either present or absent in the offer and/or answer.

If the answer crypto line does not contain an EKT parameter, then EKT MUST NOT be used for the corresponding SRTP session. Note that if the accepted crypto attribute contained a mandatory EKT parameter in the offer, and the crypto attribute in the answer does not contain an EKT parameter, then negotiation has failed (Section 5.1.3 of [RFC4568]).

If the answer crypto line contains an EKT parameter but the corresponding offered crypto line did not, or if the values don't match or are invalid, then the offerer MUST consider the crypto line invalid (see Section 7.1.3 of [RFC4568] for further operation).

The EKT parameter set is established when the answer is received, however there are a couple of special cases to consider here. First of all, if an SRTP packet containing a Full EKT Field is received prior to the answer, then the EKT parameter set is established provisionally based on the SPI included. Once the answer (which may include declarative session parameters) is received, the EKT parameter set is fully established. The second case involves receipt of multiple answers due to SIP forking. In this case, there will be multiple EKT parameter sets; one for each SRTP session. As mentioned earlier, reliable correlation of SIP dialogs to SRTP sessions requires extensions, and hence if one or more of the answers include declarative session parameters, it may be difficult to fully establish the EKT parameter set for each SRTP session. In the absence of a specific correlation mechanism, it is RECOMMENDED, that such correlation be done based on the signaled receive IP-address in the SDP and the observed source IP-address in incoming SRTP/SRTCP packets, and, if necessary, the signaled receive UDP port and the observed source UDP port.

3.6. SRTP-Specific Use Outside Offer/Answer

Security Descriptions use for SRTP is not defined outside offer/answer and hence neither does Security Descriptions with EKT.

3.7. Modifying the Session

When a media stream using the SRTP security descriptions has been established, and a new offer/answer exchange is performed, the offerer and answerer MUST follow the steps in Section 7.1.4 of [RFC4568] as well as the following steps. SDESC allows for all parameters of the session to be modified, and the EKT session parameter are no exception to that, however, there are a few additional rules to be adhered to when using EKT.

It is permissible to start a session without the use of EKT, and then subsequently start using EKT, however the converse is not. Thus, once use of EKT has been negotiated on a particular media stream, EKT MUST continue to be used on that media stream in all subsequent offer/answer exchanges.

The reason for this is that both SDESC and EKT communicate the SRTP master key with EKT communicated master keys taking precedence. Reverting back to an SDESC-controlled master key in a synchronized manner is difficult.

Once EKT is being used, the salt for the direct SRTP session MUST NOT be changed. Thus, a new offer/answer which does not create a new SRTP session (e.g., because it reuses the same IP address and port) MUST use the same salt for all crypto attributes as is currently used for the direct SRTP session.

[[Editor's Note: following paragraph would benefit from re-arranging into earlier-described steps.]]

Finally, subsequent offer/answer exchanges MUST NOT remap a given SPI value to a different EKT parameter set until 2^15 other mappings have been used within the SRTP session. In practice, this requirements is most easily met by using a monotonically increasing SPI value (modulo 2^15 and starting with zero) per direct SRTP session. Note that a direct SRTP session may span multiple SIP dialogs, and in such cases coordination of SPI values across those SIP dialogs will be required. In the simple point-to-point unicast case without translators, the requirement simply applies within each media line in the SDP. In the point-to-multipoint case, the requirement applies across all the associated SIP dialogs.

3.8. Backwards Compatibility Considerations

Backwards compatibility can be achieved in a couple of ways. First of all, Security Descriptions allows for session parameters to be prefixed with "-" to indicate that they are optional. If the answerer does not support the EKT session parameter, such optional parameters will simply be ignored. When the answer is received, absence of the parameter will indicate that EKT is not being used. Receipt of SRTP or SRTCP packets prior to receipt of such an answer will obviously be problematic (as is normally the case for Security Descriptions without EKT).

Alternatively, Security Descriptions allows for multiple crypto lines to be included for a particular media stream. Thus, two crypto lines that differ in their use of EKT parameters (presence in one, absence in the other) can be used as a way to negotiate use of EKT. When the answer is received, the accepted crypto attribute will indicate whether EKT is being used or not.

3.9. Grammar

The ABNF [RFC5234] syntax for the one new SDP Security Descriptions session parameter, EKT, comprising three parts is shown in Figure 6.

```
ekt        = "EKT=" cipher "|" key "|" spi
cipher     = cipher-ext / "AESKW_128" / "AESKW_192" / "AESKW_256"
cipher-ext = 1*64(ALPHA / DIGIT / "_")
key        = 1*(base64)    ; See Section 4 of [RFC4648]
base64     = ALPHA / DIGIT / "+" / "/" / "="
spi        = 4HEXDIG   ; See [RFC5234]
```

Figure 6: ABNF for the EKT session parameters

Using the example from Figure 6 with the EKT extensions to SDP Security Descriptions results in the following example SDP:

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
  inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20
  FEC_ORDER=FEC_SRTP EKT=AESKW_128|WWVzQUxvdmVseUVLVGtleQ|AAE0
a=crypto:2 F8_128_HMAC_SHA1_80
  inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20
  FEC_ORDER=FEC_SRTP EKT=AESKW_128|VHdvTG92ZWx5RUtUa2V5cw|AAE1
```

For legibility the SDP shows line breaks that are not present on the wire.

Figure 7: SDP Security Descriptions example with EKT

4. Use of EKT with DTLS-SRTP

This document defines an extension to DTLS-SRTP called Key Transport. The EKT with the DTLS-SRTP Key Transport enables secure transport of EKT keying material from one DTLS-SRTP peer to another. This enables those peers to process EKT keying material in SRTP (or SRTCP) and retrieve the embedded SRTP keying material. This combination of protocols is valuable because it combines the advantages of DTLS (strong authentication of the endpoint and flexibility) with the advantages of EKT (allowing secure multiparty RTP with loose coordination and efficient communication of per-source keys).

4.1. DTLS-SRTP Recap

DTLS-SRTP [RFC5764] uses an extended DTLS exchange between two peers to exchange keying material, algorithms, and parapeters for SRTP. The SRTP flow operates over the same transport as the DTLS-SRTP exchange (i.e., the same 5-tuple). DTLS-SRTP combines the performance and encryption flexibility benefits of SRTP with the flexibility and convenience of DTLS-integrated key and association management. DTLS-SRTP can be viewed in two equivalent ways: as a new key management method for SRTP, and a new RTP-specific data format for DTLS.

4.2. EKT Extensions to DTLS-SRTP

This document adds a new TLS negotiated extension called "ekt". This adds a new TLS content type, EKT, and a new negotiated extension EKT. The negotiated extension MUST only be requested in conjunction with the "use_srtp" extension (Section 3.2 of [RFC5764]). The DTLS server MUST include "dtls-srtp-ekt" in its SDP (as a session or media level attribute) and "ekt" in its TLS ServerHello message. If a DTLS client includes "ekt" in its ClientHello, but does not receive "ekt" in the ServerHello, the DTLS client MUST NOT send DTLS packets with the "ekt" content-type.

The formal description of the dtls-srtp-ekt attribute is defined by the following ABNF [RFC5234] syntax:

```
attribute = "a=dtls-srtp-ekt"
```

Using the syntax described in DTLS [RFC6347], the following structures are used:

```
                enum {
                  ekt_key(0),
                  ekt_key_ack(1),
                  ekt_key_error(254),
                  (255)
                } SRTPKeyTransportType;

                struct {
                  SRTPKeyTransportType keytrans_type;
                  uint24 length;
                  uint16 message_seq;
                  uint24 fragment_offset;
                  uint24 fragment_length;
                  select (SRTPKeyTransportType) {
                     case ekt_key:
                        EKTkey;
                   };
                } KeyTransport;

                enum {
                 RESERVED(0),
                 AESKW_128(1),
                 AESKW_192(2),
                 AESKW_256(3),
                } ektcipher;

                struct {
                  ektcipher EKT_Cipher;
                  uint EKT_Key_Value<1..256>;
                  uint EKT_Master_Salt<1..256>;
                  uint16 EKT_SPI;
                } EKTkey;
```

Figure 8: Additional TLS Data Structures

The diagram below shows a message flow of DTLS client and DTLS server using the DTLS-SRTP Key Transport extension. SRTP packets exchanged prior to the ekt_message are encrypted using the SRTP master key derived from the normal DTLS-SRTP key derivation function. After the ekt_key message, they can be encrypted using the SRTP key carried by EKT.

```
        Client                                               Server

        ClientHello + use_srtp + EKT
                                    -------->
                                     ServerHello + use_srtp + EKT
                                                     Certificate*
                                               ServerKeyExchange*
                                              CertificateRequest*
                                    <--------     ServerHelloDone
        Certificate*
        ClientKeyExchange
        CertificateVerify*
        [ChangeCipherSpec]
        Finished                    -------->
                                               [ChangeCipherSpec]
                                    <--------            Finished
        SRTP packets                <------->      SRTP packets
        SRTP packets                <------->      SRTP packets
        ekt_key                     -------->
        SRTP packets                <------->      SRTP packets
        SRTP packets                <------->      SRTP packets
```

Figure 9: Handshake Message Flow

4.3. Offer/Answer Considerations

This section describes Offer/Answer considerations for the use of EKT together with DTLS-SRTP for unicast and multicast streams. The offerer and answerer MUST follow the procedures specified in [RFC5764] as well as the following ones.

As most DTLS-SRTP processing is performed on the media channel, rather than in SDP, there is little processing performed in SDP other than informational and to redirect DTLS-SRTP to an alternate host. Advertising support for the extension is necessary in SDP because in some cases it is required to establish an SRTP call. For example, a mixer may be able to only support SRTP listeners if those listeners implement DTLS Key Transport (because it lacks the CPU cycles necessary to encrypt SRTP uniquely for each listener).

4.3.1. Generating the Initial Offer

The initial offer contains a new SDP attribute, "dtls-srtp-ekt", which contains no value. This attribute MUST only appear at the media level. This attribute indicates the offerer is capable of supporting DTLS-SRTP with EKT extensions, and indicates the desire to use the "ekt" extension during the DTLS-SRTP handshake.

An example of SDP containing the dtls-srtp-ekt attribute::

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1
  4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=dtls-srtp-ekt
```

For legibility the SDP shows line breaks that are not present on the wire.

4.3.2. Generating the Initial Answer

Upon receiving the initial offer, the presence of the dtls-srtp-ekt attribute indicates a desire to receive the EKT extension in the DTLS-SRTP handshake. DTLS messages should be constructed according to those two attributes.

If the answerer does not wish to perform EKT, it MUST NOT include a=dtls-srtp-ekt in the SDP answer, and it MUST NOT negotiate EKT during its DTLS-SRTP exchange.

Otherwise, the dtls-srtp-ekt attribute SHOULD be included in the answer, and EKT SHOULD be negotiated in the DTLS-SRTP handshake.

4.3.3. Processing the Initial Answer

The presence of the dtls-srtp-ekt attribute indicates a desire by the answerer to perform DTLS-SRTP with EKT extensions. There are two indications the remote peer does not want to do EKT: the dtls-srtp-ekt attribute is not present in the answer, or the DTLS-SRTP exchange fails to negotiate the EKT extension. If the dtls-srtp-ekt attribute is not present in the answer, the DTLS-SRTP exchange MUST NOT attempt to negotiate the EKT extension. If the dtls-srtp-ekt attribute is present in the answer but the DTLS-SRTP exchange fails to negotiate the EKT extension, EKT MUST NOT be used with that media stream.

After successful DTLS negotiation of the EKT extension, the DTLS client and server MAY exchange SRTP packets, encrypted using the KDF described in [RFC5764]. This is normal and expected, even if Key Transport was negotiated by both sides, as neither side may (yet) have a need to alter the SRTP key. However, it is also possible that one (or both) peers will immediately send an EKT packet before sending any SRTP, and also possible that SRTP, encrypted with an unknown key, may be received before the EKT packet is received.

4.3.4. Sending DTLS EKT Key Reliably

In the absence of a round trip time estimate, the DTLS ekt_key message is sent using an exponential backoff initialized to 250ms, so that if the first message is sent at time 0, the next transmissions are at 250ms, 500ms, 1000ms, and so on. If a recent round trip time estimate is available, exponential backoff is used with the first transmission at 1.5 times the round trip time estimate. In either case, re-transmission stops when ekt_key_ack or ekt_key_error message is received for the matching message_seq.

4.3.5. Modifying the Session

As DTLS-SRTP-EKT processing is done on the DTLS-SRTP channel (media channel) rather than signaling, no special processing for modifying the session is necessary.

If the initial offer and initial answer both contained EKT attributes (indicating the answerer desired to perform EKT), a subsequent offer/answer exchange MUST also contain those same EKT attributes. If not, operation is undefined and the sesion MAY be terminated. If the initial offer and answer failed to negotiate EKT (that is, the answer did not contain EKT attributes), EKT negotiation failed and a subsequent offer SHOULD NOT include EKT attributes.

5. Use of EKT with MIKEY

The advantages outlined in Section 1 are useful in some scenarios in which MIKEY is used to establish SRTP sessions. In this section, we briefly review MIKEY and related work, and discuss these scenarios.

An SRTP sender or a group controller can use MIKEY to establish a SRTP cryptographic context. This capability includes the distribution of a TEK generation key (TGK) or the TEK itself, security policy payload, crypto session bundle ID (CSB_ID) and a crypto session ID (CS_ID). The TEK directly maps to an SRTP master key, whereas the TGK is used along with the CSB_ID and a CS_ID to generate a TEK. The CS_ID is used to generate multiple TEKs (SRTP master keys) from a single TGK. For a media stream in SDP, MIKEY allocates two consecutive numbers for the crypto session IDs, so that each direction uses a different SRTP master key (see [RFC4567]).

The MIKEY specification [RFC3830] defines three modes to exchange keys, associated parameters and to protect the MIKEY message: pre-shared key, public-key encryption and Diffie-Hellman key exchange. In the first two modes the MIKEY initiator only chooses and distributes the TGK or TEK, whereas in the third mode both MIKEY entities (the initiator and responder) contribute to the keys. All three MIKEY modes have in common that for establishing a SRTP session the exchanged key is valid for the send and receive direction. Especially for group communications it is desirable to update the SRTP master key individually per direction. EKT provides this property by distributing the SRTP master key within the SRTP/SRTCP packet.

MIKEY already supports synchronization of ROC values between the MIKEY initiator and responder. The SSRC / ROC value pair is part of the MIKEY Common Header payload. This allows providing the current ROC value to late joiners of a session. However, in some scenarios a key management based ROC synchronization is not sufficient. For example, in mobile and wireless environments, members may go in and out of coverage and may miss a sequence number overrun. In point-to-multipoint translator scenarios it is desirable to not require the group controller to track the ROC values of each member, but to provide the ROC value by the originator of the SRTP packet. A better alternative to synchronize the ROC values is to send them directly via SRTP/SRTCP as EKT does. A separate SRTP extension [RFC4771] includes the ROC in a modified authentication tag but that extension does not support updating the SRTP master key.

Besides the ROC, MIKEY synchronizes also the SSRC values of the SRTP streams. Each sender of a stream sends the associated SSRC within the MIKEY message to the other party. If an SRTP session participant starts a new SRTP source (SSRC) or a new participant is added to a group, subsequent SDP offer/answer and MIKEY exchanges are necessary to update the SSRC values. EKT improves these scenarios by updating the keys and SSRC values without coordination on the signaling channel. With EKT, SRTP can handle early media, since the EKT SPI allows the receiver to identify the cryptographic keys and parameters used by the source.

The MIKEY specification [RFC3830] suggests the use of unicast for rekeying. This method does not scale well to large groups or interactive groups. The EKT extension of SRTP/SRTCP provides a solution for rekeying the SRTP master key and for ROC/SSRC synchronization. EKT is not a substitution for MIKEY, but rather a complementary addition to address the above described limitations of MIKEY.

In the next section we provide an extension to MIKEY for support of EKT. EKT can be used only with the pre-shared key or public-key encryption MIKEY mode of [RFC3830]. The Diffie-Hellman exchange mode is not suitable in conjunction with EKT, because it is not possible to establish one common EKT key over multiple EKT entities. Additional MIKEY modes specified in separate documents are not considered for EKT.

5.1. EKT Extensions to MIKEY

In order to use EKT with MIKEY, the EKT cipher, EKT key and EKT SPI is negotiated in the MIKEY message exchange.

The following parameters are added to the MIKEY Security Protocol Parameters namespace ([RFC3830], Section 6.10.1). (TBD will be requested from IANA [NOTE TO RFC EDITOR])

Type	Meaning	Possible values
TBD	EKT cipher	see below
TBD	EKT SPI	a 15-bit value

Figure 10: MIKEY Security Protocol Parameters

EKT cipher	Value
(reserved)	0
AESKW_128	1
AESKW_192	2
AESKW_256	3

Figure 11: EKT Cipher Parameters

EKT_Key is transported in the MIKEY KEMAC payload within one separate Key Data sub-payload. As specified in Section 6.2 of [RFC3830], the KEMAC payload carries the TEK Generation Key (TGK) or the Traffic Encryption Key (TEK). One or more TGKs or TEKs are carried in individual Key Data sub-payloads within the KEMAC payload. The KEMAC payload is encrypted as part of MIKEY. The Key Data sub- payload, specified in Section 6.13 of [RFC3830], has the following format:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload  | Type  |  KV   | Key data length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                           Key data                            :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
: Salt length (optional)        ! Salt data (optional)          :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                          KV data (optional)                   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Key Data Sub-Payload of MIKEY

These fields are described below:

Type: 4 bits in length, indicates the type of key included in the payload. We define Type = TBD (will be requested from IANA [NOTE TO RFC EDITOR]) to indicate transport of the EKT key.

KV: (4 bits): indicates the type of key validity period specified. KV=1 is currently specified as an SPI. We use that value to indicate the KV data contains the EKT_SPI for the key type EKT_Key. KV data would be 16 bits in length, but it is also possible to interpret the length from the 'Key data len' field. KV data MUST be present for the key type EKT_Key when KV=1.

Salt length, Salt Data: These optional fields SHOULD be omitted for the key type EKT_Key, if the SRTP master salt is already present in the TGK or TEK Key Data sub-payload. The EKT_Key sub-payload MUST contain a SRTP master salt, if the SRTP master salt is not already present in the TGK or TEK Key Data sub-payload.

KV Data: length determined by Key Data Length field.

5.2. Offer/Answer Considerations

This section describes Offer/Answer considerations for the use of EKT together with MIKEY for unicast streams. The offerer and answerer MUST follow the procedures specified in [RFC3830] and [RFC4567] as well as the following ones.

5.2.1. Generating the Initial Offer

If it is intended to use MIKEY together with EKT, the offerer MUST include at least one MIKEY key-mgmt attribute with one EKT_Key Key Data sub-payload and the SRTP Security Policy payload (SP) with the policy parameter EKT SPI. The policy parameter EKT Cipher is

OPTIONAL, The default value is "AESKW_128" in accordance with Section 2.3.1. MIKEY can be used on session or media level. On session level, MIKEY provides the keys for multiple SRTP sessions in the SDP offer. The EKT SPI references a EKT parameter set including the Secure RTP parameters as specified in Section 8.2 in [RFC3711]. If MIKEY is used on session level, it is only possible to use one EKT SPI value. Therefore, the session-level MIKEY message MUST contain one SRTP Security Policy payload only, which is valid for all related SRTP media lines. If MIKEY is used on media level, different SRTP Security Policy parameters (and consequently different EKT SPI values) can be used for each media line. If MIKEY is used on session and media level, the media level content overrides the session level content.

EKT requires a single shared SRTP master salt between all participants in the direct SRTP session. If a MIKEY key-mgmt attribute contains more than one TGK or TEK Key Data sub-payload, all the sub-payloads MUST contain the same master salt value. Consequently, the EKT_Key Key Data sub-payload MAY also contain the same salt or MAY omit the salt value. If the SRTP master salt is not present in the TGK and TEK Key Data sub-payloads, the EKT_Key sub-payload MUST contain a master salt.

5.2.2. Generating the Initial Answer

For each media line in the offer using MIKEY, provided on session and/or on media level, the answerer examines the related MIKEY key-mgmt attributes for the presence of EKT parameters. In order to accept the offered key-mgmt attribute, the MIKEY message MUST contain one EKT_Key Key Data sub-payload and the SRTP Security Policy payload with policy parameter EKT SPI. The answerer examines also the existence of a SRTP master salt in the TGK/TEK and/or the EKT_Key sub-payloads. If multiple salts are available, all values MUST be equal. If the salt values differ or no salt is present, the key-mgmt attribute MUST be considered as invalid.

The MIKEY responder message in the SDP answer does not contain a MIKEY KEMAC or Security Policy payload and consequently does not contain any EKT parameters. If a key-mgmt attribute for a media line was accepted by the answerer, the EKT parameter set of the offerer is valid for both directions of the SRTP session.

5.2.3. Processing the Initial Answer

On reception of the answer, the offerer examines if EKT has been accepted for the offered media lines. If a MIKEY key-mgmt attribute is received containing a valid MIKEY responder message, EKT has been successfully negotiated. On receipt of a MIKEY error message, EKT negotiation has failed. For example, this may happen if an EKT extended MIKEY initiator message is sent to a MIKEY entity not supporting EKT. A MIKEY error code 'Invalid SPpar' or 'Invalid DT' is returned to indicate that the EKT parameters (EKT Cipher and EKT SPI) in the SRTP Security Policy payload or the EKT_Key sub-payload is not supported. In this case, the offerer may send a second SDP offer with a MIKEY key-mgmt attribute without the additional EKT extensions.

This behavior can be improved by offering two key-mgmt SDP attributes. One attribute offers MIKEY with SRTP and EKT and the other attribute offers MIKEY with SRTP without EKT.

5.2.4. Modifying the Session

Once an SRTP stream has been established, a new offer/answer exchange can modify the session including the EKT parameters. If the EKT key or EKT cipher is modified (i.e., a new EKT parameter set is created) the offerer MUST also provide a new EKT SPI value. The offerer MUST NOT remap an existing EKT SPI value to a new EKT parameter set. Similar, a modification of the SRTP Security Policy leads to a new EKT parameter set and requires a fresh EKT SPI, even if the EKT key or cipher did not change.

Once EKT is being used, the SRTP master salt for the SRTP session MUST NOT be changed. The salt in the Key Data sub-payloads within the subsequent offers MUST be the same as the one already used.

After EKT has been successfully negotiated for a session and an SRTP master key has been transported by EKT, it is difficult to switch back to a pure MIKEY based key exchange in a synchronized way. Therefore, once EKT is being used for a session, EKT MUST be used also in all subsequent offer/answer exchanges for that session.

6. Using EKT for Interoperability between Key Management Systems

A media gateway (MGW) can provide interoperability between an SRTP-EKT endpoint and a non-EKT SRTP endpoint. When doing this function, the MGW can perform non-cryptographic transformations on SRTP packets outlined above. However, there are some uses of cryptography that will be required for that gateway. If a new SRTP master key is communicated to the MGW (via EKT from the EKT leg, or via Security Descriptions without EKT from the Security Descriptions leg), the MGW needs to convert that information for the other leg, and that process will incur some cryptographic operations. Specifically, if the new key arrived via EKT, the key must be decrypted and then sent in Security Descriptions (e.g., as a SIP re-INVITE); likewise, if a new key arrives via Security Descriptions that must be encrypted via EKT and sent in SRTP/SRTCP.

Additional non-normative information can be found in Appendix A.

7. Design Rationale

From [RFC3550], a primary function of RTCP is to carry the CNAME, a "persistent transport-level identifier for an RTP source" since "receivers require the CNAME to keep track of each participant." EKT works in much the same way but uses SRTP to carry information needed for the proper processing of the SRTP traffic.

With EKT, SRTP gains the ability to synchronize the creation of cryptographic contexts across all of the participants in a single session. This feature provides some, but not all, of the functionality that is present in IKE phase two (but not phase one). Importantly, EKT does not provide a way to indicate SRTP options.

With EKT, external signaling mechanisms provide the SRTP options and the EKT Key, but need not provide the key(s) for each individual SRTP source. EKT provides a separation between the signaling mechanisms and the details of SRTP. The signaling system need not coordinate all SRTP streams, nor predict in advance how many sources will be present, nor communicate SRTP-level information (e.g., rollover counters) of current sessions.

EKT is especially useful for multi-party sessions, and for the case where multiple RTP sessions are sent to the same destination transport address (see the example in the definition of "RTP session" in [RFC3550]). A SIP offer that is forked in parallel (sent to multiple endpoints at the same time) can cause multiple RTP sessions to be sent to the same transport address, making EKT useful for use with SIP.

EKT can also be used in conjunction with a scalable group-key management system like GDOI [RFC6407]. In such a combination GDOI would provide a secure entity authentication method for group members, and a scalable way to revoke group membership; by itself, EKT does not attempt to provide either capability.

EKT carries the encrypted key in a new SRTP field (at the end of the SRTP packet). This maintains compatibility with the existing SRTP specification by defining a new crypto function that incorporates the encrypted key, and a new authentication transform to provide implicit authentication of the encrypted key.

The main motivation for the use of the variable-length EKT format is bandwidth conservation. When EKT is sent over SRTP, there will be a loss of (usable) bandwidth due to the additional EKT bytes in each RTP packet. For some applications, this bandwidth loss is significant.

7.1. Alternatives

In its current design, EKT requires that the Master Salt be established out of band. That requirement is undesirable. In an offer/answer environment, it forces the answerer to re-use the same Master Salt value used by the offerer. The Master Salt value could be carried in EKT packets though that would consume yet more bandwidth.

In some scenarios, two SRTP sessions may be combined into a single session. When using EKT in such sessions, it is desirable to have an SPI value that is larger than 15 bits, so that collisions between SPI values in use in the two different sessions are unlikely (since each collision would confuse the members of one of the sessions).

An alternative that addresses both of these needs is as follows: the SPI value can be lengthed from 15 bits to 63 bits, and the Master Salt can be identical to, or constructed from, the SPI value. SRTP conventionally uses a 14-byte Master Salt, but shorter values are acceptable. This alternative would add six bytes to each EKT packet; that overhead may be a reasonable tradeoff for addressing the problems outlined above. This is considered too high a bandwidth penalty.

8. Security Considerations

EKT inherits the security properties of the SRTP keying it uses: Security Descriptions, DTLS-SRTP, or MIKEY.

With EKT, each SRTP sender and receiver MUST generate distinct SRTP master keys. This property avoids any security concern over the re-use of keys, by empowering the SRTP layer to create keys on demand. Note that the inputs of EKT are the same as for SRTP with key-sharing: a single key is provided to protect an entire SRTP session. However, EKT remains secure even in the absence of out-of-band coordination of SSRCs, and even when SSRC values collide.

The EKT Cipher includes its own authentication/integrity check. For an attacker to successfully forge a full EKT packet, it would need to defeat the authentication mechanisms of both the EKT Cipher and the SRTP authentication mechanism.

The presence of the SSRC in the EKT_Plaintext ensures that an attacker cannot substitute an EKT_Ciphertext from one SRTP stream into another SRTP stream.

An attacker who strips a Full_EKT_Field from an SRTP packet may prevent the intended receiver of that packet from being able to decrypt it. This is a minor denial of service vulnerability. Similarly, an attacker who adds a Full_EKT_Field can disrupt service.

An attacker could send packets containing either Short EKT Field or Full EKT Field, in an attempt to consume additional CPU resources of the receiving system. In the case of the Short EKT Field, this field is stripped and normal SRTP or SRTCP processing is performed. In the case of the Full EKT Field, the attacker would have to have guessed or otherwise determined the SPI being used by the receiving system. If an invalid SPI is provided by the attacker, processing stops. If a valid SPI is provided by the attacker, the receiving system will decrypt the EKT ciphertext and return an authentication failure (Step 3 of Section 2.2.2).

EKT can rekey an SRTP stream until the SRTP rollover counter (ROC) needs to roll over. EKT does not extend SRTP's rollover counter (ROC), and like SRTP itself EKT cannot properly handle a ROC rollover. Thus even if using EKT, new (master or session) keys need to be established after 2^48 packets are transmitted in a single SRTP stream as described in Section 3.3.1 of [RFC3711]. Due to the relatively low packet rates of typical RTP sessions, this is not expected to be a burden.

The confidentiality, integrity, and authentication of the EKT cipher MUST be at least as strong as the SRTP cipher.

Part of the EKT_Plaintext is known, or easily guessable to an attacker. Thus, the EKT Cipher MUST resist known plaintext attacks. In practice, this requirement does not impose any restrictions on our choices, since the ciphers in use provide high security even when much plaintext is known.

An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen. An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen and adversaries that can query both the encryption and decryption functions adaptively.

9. IANA Considerations

IANA is requested to register EKT from Section 3.9 into the Session Description Protocol (SDP) Security Descriptions [iana-sdp-sdesc] registry for "SRTP Session Parameters".

IANA is requested to register the following new attributes into the SDP Attributes registry [iana-sdp-attr].

Attribute name: dtls-srtp-ekt

Long form name: DTLS-SRTP with EKT

Type of attribute: Media-level

Subject to charset: No

Purpose: Indicates support for DTLS-SRTP with EKT

Appropriate values: No values

Contact name: Dan Wing, dwing@cisco.com

We request the following IANA assignments from the existing [iana-mikey] name spaces in the IETF consensus range (0-240) [RFC3830]:

- o From the Key Data payload name spaces, a value to indicate the type as the 'EKT_Key'.

Furthermore, we need the following two new IANA registries created, populated with the initial values in this document. New values for both of these registries can be defined via Specification Required [RFC5226].

- o EKT parameter type, initially populated with the list from Figure 10

- o EKT cipher, initially populated with the list from Figure 11

10. Acknowledgements

Thanks to Lakshminath Dondeti for assistance with earlier versions of this document. Thanks to Kai Fischer for writing the MIKEY section.

Thanks to Nermeen Ismail, Eddy Lem,Rob Raymond, and Yi Cheng for fruitful discussions and comments. Thanks to Felix Wyss for his review and comments regarding ciphers. Thanks to Michael Peck for his review. Thanks to Magnus Westerlund for his review. Thanks to Michael Peck and Jonathan Lennox for their review comments.

11. References

11.1. Normative References

[FIPS197] National Institute of Standards and Technology (NIST), "The Advanced Encryption Standard (AES)", FIPS-197 Federal Information Processing Standard, November 2001.

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[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.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4563] Carrara, E., Lehtovirta, V., and K. Norrman, "The Key ID Information Type for the General Extension Payload in Multimedia Internet KEYing (MIKEY)", RFC 4563, June 2006.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

11.2. Informative References

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

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

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, January 2007.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, September 2009.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, October 2011.

[iana-mikey] IANA, , "Multimedia Internet KEYing (Mikey) Payload Name Spaces", 2011, <http://www.iana.org/assignments/mikey-payloads/mikey-payloads.xhtml>.

[iana-sdp-attr] IANA, , "SDP Parameters", 2011, <http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>.

[iana-sdp-sdesc]
IANA, , "Session Description Protocol (SDP) Security Descriptions: SRTP Session Parameters", 2011, <http://www.iana.org/assignments/sdp-security-descriptions/sdp-security-descriptions.xml#sdp-security-descriptions-4>.

Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with Security Descriptions

Today, SDP Security Descriptions [RFC4568] is used for distributing SRTP keys in several different IP PBX systems. The IP PBX systems are typically used within a single enterprise. A Session Border Controller is a reasonable solution to interwork between Security Descriptions in one network and DTLS-SRTP in another network. For example, a mobile operator (or an Enterprise) could operate Security Descriptions within their network and DTLS-SRTP towards the Internet.

However, due to the way Security Descriptions and DTLS-SRTP manage their SRTP keys, such an SBC has to authenticate, decrypt, re-encrypt, and re-authenticate the SRTP (and SRTCP) packets in one direction, as shown in Figure 13, below. This is computationally expensive.

```
    RFC4568 endpoint           SBC           DTLS-SRTP endpoint
             |                      |                        |
     1.      |---key=A------------->|                        |
     2.      |                      |<-DTLS-SRTP handshake-> |
     3.      |<--key=B--------------|                        |
     4.      |                      |<--SRTP, encrypted w/B- |
     5.      |<-SRTP, encrypted w/B-|                        |
     6.      |-SRTP, encrypted w/A->|                        |
     7.      |            (decrypt, re-encrypt)              |
     8.      |                      |-SRTP, encrypted w/C--> |
             |                      |                        |
```

Figure 13: Interworking Security Descriptions and DTLS-SRTP

The message flow is as follows (similar steps occur with SRTCP):

1. The Security Descriptions [RFC4568] endpoint discloses its SRTP key to the SBC, using a=crypto in its SDP.

2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).

3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint (using a=crypto). (There is no way, with DTLS-SRTP, to communicate the Security Descriptions key to the DTLS-SRTP key endpoint.)

4. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.

5. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was already communicated in step 3.

6. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.

7. The SBC has to authenticate and decrypt the SRTP packet (using key A), and re-encrypt it and generate an HMAC (using key C).

8. The SBC sends the new SRTP packet.

If EKT is deployed on the DTLS-SRTP endpoints, EKT helps to avoid the computationally expensive operation so the SBC does not need to perform any per-packet operations on the SRTP (or SRTCP) packets in either direction. With EKT the SBC can simply forward the SRTP (and SRTCP) packets in both directions without per-packet HMAC or cryptographic operations.

To accomplish this interworking, DTLS-SRTP EKT must be supported on the DTLS-SRTP endpoint, which allows the SBC to transport the Security Description key to the EKT endpoint and send the DTLS-SRTP key to the Security Descriptions endpoint. This works equally well for both incoming and outgoing calls. An abbreviated message flow is shown in Figure 14, below.

```
RFC4568 endpoint           SBC          DTLS-SRTP endpoint
        |                     |                        |
  1.    |---key=A------------>|                        |
  2.    |                     |<-DTLS-SRTP handshake-> |
  3.    |<--key=B-------------|                        |
  4.    |                     |--ekt:A---------------> |
  5.    |                     |<--SRTP, encrypted w/B- |
  5.    |<-SRTP, encrypted w/B-|                       |
  6.    |-SRTP, encrypted w/A->|                       |
  7.    |                     |-SRTP, encrypted w/A--> |
        |                     |                        |
```

Figure 14: Interworking Security Descriptions and EKT

The message flow is as follows (similar steps occur with SRTCP):

1. Security Descriptions endpoint discloses its SRTP key to the SBC (a=crypto).

2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).

3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint.

4. The SBC sends an EKT packet indicating that SRTP will be encrypted with 'key A' towards the DTLS-SRTP endpoint.

5. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.

6. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was communicated in step 3.

7. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.

8. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key A) was communicated in step 4.

Authors' Addresses

John Mattsson (editor)
Ericsson AB
SE-164 80 Stockholm
Sweden

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

David A. McGrew
Cisco Systems, Inc.
510 McCarthy Blvd.
Milpitas, CA 95035
US

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm

Dan Wing
Cisco Systems, Inc.
510 McCarthy Blvd.
Milpitas, CA 95035
US

Phone: (408) 853 4197
Email: dwing@cisco.com

Flemming Andreason
Cisco Systems, Inc.
499 Thornall Street
Edison, NJ 08837
US

Email: fandreas@cisco.com

Network Working Group P. Jones (Ed.)
Internet Draft N. Ismail
Intended status: Standards Track D. Benham
Expires: September 7, 2015 Cisco Systems
March 7, 2015

A Solution Framework for Private Media in a Switched Conferencing
draft-jones-avtcore-private-media-framework-01

Abstract

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Switched conferencing is an increasingly popular model for multimedia conferences with multiple participants using a combination of audio, video, text, and other media types. With this model, real-time media flows from conference participants are not mixed, transcoded, transrated, recomposed, or otherwise manipulated on the conference server, as might be the case with a traditional multipoint control unit (MCU). Instead, media flows transmitted by conference participants are simply forwarded by the switching conference server to each of the other participants, selectively forwarding flows based on voice activity detection or other criteria. In some instances, the switching conference server may make limited modifications to RTP [RFC3550] headers, for example, but the actual media content (e.g., voice or video data) is unaltered.

An advantage of switched conferencing is that conference servers can be deployed on general-purpose computing hardware, as there is no need for the specialized hardware required to manipulate media flows that one finds on a traditional hardware MCU. This, in turn, means that it is possible to deploy switching conference servers in virtualized environments, including private and public clouds.

However, deploying conference resource in a cloud environment may introduce a higher security risk. Whereas traditional conference servers were usually deployed in private networks that were protected from public access by firewalls, cloud-based conference resources might be viewed as less secure since they are not always physically controlled by those who use the hardware. Additionally, there are usually several ports open to the public in cloud deployments, most significantly being ports where the administrator can log in to make configuration changes, install software updates, and so on.

Recognizing the need to improve the way in which media confidentiality is ensured, requirements for private media were specified in [I.D-draft-jones-avtcore-private-media-reqts]. Attempting to meet those requirements, this document defines a solution framework wherein privacy is ensured by making it impossible for a switching conference server to gain access to keys needed to decrypt or authenticate the actual media content sent between conference participants. At the same time, the framework allows for the switching conference server to modify certain RTP headers; add, remove, encrypt, or decrypt RTP header extensions; and encrypt and decrypt RTCP packets. The framework also prevents replay attacks by authenticating each packet transmitted between a given participant and the switching conference server by using a key that is independent from the media encryption and authentication key(s) and is unique to the participating endpoint and the switching conference server.

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

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Private Media Trust Model

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

4. Solution Framework Overview

The purpose for this framework is to define a means through which media privacy can be ensured when communicating within a switched conferencing environment. This framework specifies the re-use of several technologies, including SRTP [RFC3711], EKT [I.D-draft-ietf-avtcore-srtp-ekt], and DTLS-SRTP [RFC5764].

4.1. Switching Conference Server Behavior

Before going into the specifics of how media privacy is ensured, first consider Figure 1 below depicting the behavior of a switching conferencing server forwarding media between participants.

```
                      +--------------------+
+---+ --{A}--> |                    | <-{C}--- +---+
| A | <-{B}--- |Switching Conference| --{A}--> | C |
|   | <-{C}--- |       Server       | --{B}--> |   |
+---+ <-{D}--- |                    | --{D}--> +---+
               |       Packet       |
+---+ --{B}--> |   Authentication   | <-{D}--- +---+
| B | <-{A}--- |                    | --{A}--> | D |
|   | <-{C}--- |                    | --{B}--> |   |
+---+ <-{D}--- |   Media Privacy    | --{C}--> +---+
               +--------------------+
```

Figure 1 - Switching Conference Server

In the above figure, each of the participating endpoints sends media to the switching conference server where each flow is protected using the security mechanisms that will be discussed later. Each endpoint then receives media from each of the other participants in the conference. Importantly, the switching conference server is unable to decrypt the media content or modify media content without being detected by receiving endpoints.

The framework does not require, however, for each of the participant flows to be transmitted to every other endpoint in the conference. In many situations, the switching conference server will transmit only a subset of the media flows to each participant. This might be to restrict the bandwidth usage, provide a primary video flow and thumbnail flows to single-screen video endpoints, etc.

The following two diagrams and corresponding explanatory text are for illustrative purposes to describe one possible operational mode.

```
                     +--------------------+
        +---+ --{A}--> |                    | <--{C}-- +---+
        | A |          |Switching Conference|          | C |*
        +---+ <-{C}--- |       Server       | ---{A}-> +---+
                       |                    |
        +---+ --{B}--> |                    | <--{D}-- +---+
        | B |          |                    |          | D |
        +---+ <-{C}--- |                    | ---{C}-> +---+
                       +--------------------+
```

Figure 2 - Endpoint "C" is the Active Speaker

As depicted in Figure 2, each of the endpoints in the conference is receiving a single flow. In particular, all but one endpoints are receiving media flows from endpoint "C", the current active speaker. Endpoint "C" is receiving media from endpoint "A", the former active speaker.

```
                       +--------------------+
        +---+ --{A}--> |                    | <--{C}-- +---+
        | A |          |Switching Conference|          | C |
        +---+ <-{B}--- |       Server       | ---{B}-> +---+
                       |                    |
        +---+ --{B}--> |                    | <--{D}-- +---+
       *| B |          |                    |          | D |
        +---+ <-{C}--- |                    | ---{B}-> +---+
                       +--------------------+
```

Figure 3 - Endpoint "B" is the Active Speaker

When the active speaker transitions, so do the video flows. As depicted in Figure 3, the active speaker transitions from "C" to "B". Now, each of the endpoints receives a copy of the media flows from "B", while "B" receives the media flow from "C", the former active speaker.

How many flows and what type of flows a switching conference server transmits to a receiving endpoint are outside the scope of this document.

4.2. End-to-End Media Privacy

To ensure the confidentiality of RTP media packets, endpoints utilize EKT keys known to conference participants to encrypt the media content of the RTP packet (i.e., the RTP payload) using authenticated encryption. These keys may change from time-to-time for various reasons, such as when a new conference participant joins a conference or when a conference participant leaves a conference. When it is decided that a conference is to be re-keyed is outside the scope of this document, but it is important that an unstrusted switching conference server is never given access to those keys.

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

4.3. Hop-by-Hop Operations

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

It is expected that the conference servers may find it necessary to change certain parts of the RTP packet header, add or remove RTP header extensions, etc. By using hop-by-hop authentication, the switching media server is given liberty to change certain values present in the RTP header, such as the payload type value.

If there is a desire to encrypt RTP header extensions, an encryption key is derived from the hop-by-hop SRTP master key to encrypt header extensions as per [RFC6904]. This will give the switching conference server visibility into header extensions, such as the one used to determine audio level [RFC6464] of conference participants. Note that allowing RTP header extensions to be encrypted requires that all hops decrypt and re-encrypt any encrypted header extensions.

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

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

5. Media Packet Format

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

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
       |V=2|P|X|  CC   |M|     PT      |       sequence number         | |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
       |                           timestamp                           | |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
       |           synchronization source (SSRC) identifier            | |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
       |            contributing source (CSRC) identifiers             | |
       |                               ....                            | |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
       |                   RTP extension (OPTIONAL*)                   | |
     +>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
     | |                          payload  ...                         | |
     | |                               +-------------------------------+ |
     | |                               | RTP padding   | RTP pad count | |
     +>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
     | |                           SRTP ROC                            | |
     | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
     | |                     EKT ciphertext ...                        | |
     | |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
     | |                               |  Security Parameter Index   |1| |
     | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
     | :                authentication tag (MANDATORY)                 : |
     | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
     |                                                                   |
     +-- Authenticated Encryption                 Authenticated Portion --+
         End-to-End                               using Hop-by-Hop Key

         * Header extensions are optionally Encrypted Hop-by-Hop
```

Figure 4 - Private Media SRTP Packet

The rollover counter value is shown and transmitted as plaintext. This is necessary since a switching conference server may not transmit media from one "silent" participant to another participant in the conference for a long period of time. When media from that "silent" participant is later sent to that other participant, the receiving participant would not otherwise know the value of the rollover counter. Further, this value is needed so that the correct authentication tag can be generated hop-by-hop.

The EKT field shown in Figure 4 is the "Full EKT Field". The "Short EKT Field" may also be present in its place.

6. SRTP Cryptographic Context

For any given media source identified by its SSRC, there is a single SRTP cryptographic context as described in Section 3.2 of [RFC3711] used in this framework. However, this framework extends the parameter set of the cryptographic context by adding an identifier for the end-to-end authenticated encryption algorithm. That parameter has associated with it an SRTP master key, and as outlined in Section 3.2.1, other associated values that relate to the master key (e.g., master salt and key length values). For AES-CCM, there will also be an associated "Tag_Size_Flag" value (see [I.D-draft-ietf-avtcore-srtp-aes-gcm]).

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

7. Cryptographic Operations

7.1. Hop-by-Hop Authentication and Optional Encryption

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

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

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

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

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

This section covers the encryption and authentication of the RTP payload (i.e., media content) using the SRTP master key(s) derived from the EKT Key(s) by the endpoints communicating in a switched conferencing environment.

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

The procedures followed to encrypt the payload are those described in [I.D-draft-ietf-avtcore-srtp-aes-gcm], except that the associated data used with those algorithms specified in Section 9.2 is redefined as follows:

Associated Data: The version V (2 bits), padding flag P (1 bit), the sequence number (16 bits), timestamp (32 bits), and SSRC (32 bits).

The authentication tag for the end-to-end encrypted payload immediately follows the encrypted payload in the packet format.

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

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

8. Key Exchange

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

To facilitate key exchange, we utilize DTLS-SRTP and procedures defined in EKT. We will elaborate on this further in the following sub-sections.

8.1. Session Signaling

The session signaling protocol is not significant to this specification, since the call processing functions are untrusted. Signaling might be via SIP [RFC3261] or a proprietary signaling between a browser and a server, as examples. What is important is that the signaling convey, in some manner, the fingerprint of the endpoint’s certificate that will be used with DTLS-SRTP. For the sake of providing a more concrete discussion, we will assume SIP is used and SDP [RFC4566] conveys the fingerprint information as per [RFC5763].

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

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

The Key Management Function (KMF) needs to receive information about the call. This might be performed via an interface between the endpoint and the KMF, the call processing function and the KMF, or it might be via a signaling interface between the switching conference server and the KMF (see Error! Reference source not found.). Regardless, it is important that the endpoint's certificate fingerprint and a participant identifier (a random value created by the endpoint and provided to the KMF for each RTP session) are securely conveyed to the KMF. The client certificate and participant identifier will allow the KMF to associate the DTLS connection to the specific endpoint and RTP session for the conference. The endpoint to KMF information exchange is outside the scope of this document.

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

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

[Editor's Note: How would an endpoint that is outside an enterprise domain (e.g., an associate at another company) be able to interact with the enterprise KMF? It might be necessary to have a trusted call processing entity that signs messages that the foreign endpoint can validate so that it knows that it can trust the certificate fingerprint of the KMF.]

8.2. Negotiating SRTP Protection Profiles and Key Exchange

8.2.1. Endpoint and KMF

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

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

```
                  Conference
       +-----+    Server / KMF +-----------------------------+
       |     |    Interface    | Switching Conference Server |
       |     |<--------------->|                             |
       |     |                 |                             |
       | KMF |<--------------->|<-------------+ (Tunnels     |
       |     |    DTLS-        |              v  DTLS-SRTP)  |
       +-----+    SRTP         +-----------------------------+
                  Tunnel                      ^
                                              | DTLS-SRTP
                                              |
                                              v
                                         +----------+
                                         | Endpoint |
                                         +----------+
```

Figure 5 - DTLS-SRTP Tunneled to KMF

Through this tunneled DTLS-SRTP exchange, an EKT master key and EKT master salt are conveyed from the KMF to the endpoint, which the endpoint will use when deriving SRTP keys and encrypt and authenticate the media content in SRTP packets. The endpoint does not transmit media encryption keys to the KMF. The endpoint will follow the procedures specified in the EKT specification to generate an SRTP master key and convey this information to conference participants periodically (and anytime an I-Frame is explicitly requested) via the "Full EKT Field". [Editor’s note: we are proposing changes to the EKT draft that will include the ROC separated from the EKT Ciphertext. Additionally, we need a mechanism to negotiate SRTP Protection Profiles for the end-to-end encryption/authentication. This might be an extension to EKT, a new extension, or even an application-layer exchange over the DTLS connection to the KMF.]

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

The DTLS-SRTP procedures will result in the determination of an SRTP master key and master salt, along with an SRTP Protection Profile. This information is used for the hop-by-hop operations. [Editor's note: We could use DTLS-SRTP only to negotiate the SRTP Protection Profiles and then introduce a new extension to allow the KMF to send out the hop-by-hop key and salt to both the endpoint and conference server. Open to alternative suggestions from the workgroup.]

During the lifetime of the conference, conference participants may come and go. During those events, the KMF will send a new EKT message to clients providing a new EKT key to use from that point forward.

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

Summary of what we need to realize this framework:

- Endpoint must securely convey its certificate information to the KMF and negotiate a participant identifier (e.g., a UUID securely conveyed, but need not be encrypted) before a connection to the conference server is attempted.

- A means through EKT or another extension to negotiate the SRTP security profiles for end-to-end encryption/authentication

- A means through EKT or another extension of sending the participant identifier (the participant identifier could implicitly identify the conference)

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

- A means of conveying per-hop SRTP master key and salt information to the switching conference server

To help in understanding better the sequence of messages, consider the following figure:

```
                                                      Conference
   Endpoint                    KMF                     Server
      |                         |                          |
      | External Signaling      |                          |
      | To exchange Cert and    |                          |
      | and participant ID      |                          |
      | ----------------------> |                          |
      |                         |                          |
      | DTLS connection         |                          |
      | -------------------------------------------------> | \
      |                         | <======================= | /
      |                         |              DTLS Tunnel |
      |                         |                          |
      | For brevity, we use === for tunneled DTLS messages to KMF
      |                         |                          |
      | DTLS-SRTP and EKT       |                          |
      | ======================> |                          |
      |  (Participant ID, cert, |                          |
      |   security profiles,    |                          |
      |   etc.)                 |                          |
      |                         |                          |
      | <=====================> |                          |
      |         SRTP Master key | -----------------------> |
      |     and salt determined | SRTP Master keys         |
      |          for hop-by-hop | and salts conveyed       |
      |                         | for hop-by-hop           |
      | <====================== | (interface and           |
      |        EKT Key conveyed |  endpoint/conference     |
      |          for end-to-end |  association TBD)        |
      |                         |                          |
      |                         |                          |
```

Figure 6 - Key Exchange Procedure

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

8.2.2. Switching Conference Server and KMF

[Editor's Note: there must be an interface between the switching conference server and the KMF so that cipher suites and key information can be conveyed for each participant in each conference for hop-by-hop operations. This interface is out of scope for this document.]

9. Changing Media Forwarded and EKT Field

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

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

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

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

A switching conference server may forward a single audio and video flow to a receiver, or it may forward multiple flows. The number of media flows very much depends on the capabilities of the receiving device. How the number of media flows to forward is determined or negotiated is outside the scope of this document.

To aid in determining when to transition the active speaker's audio or video, endpoints MUST implement [RFC6464] in order to provide a hint to the switching media server as to which endpoint should be designated as the one of the active speaker(s).

10. IANA Considerations

There are no IANA considerations for this document.

11. Security Considerations

[TBD]

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.

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

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

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

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

[I.D-draft-ietf-avtcore-srtp-aes-gcm] McGrew, D. and K. Igoe, "AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP)", Work in Progress, July 2014.

[RFC5763] Fischl, J. Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.

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

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

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

12.2. Informative References

[I.D-draft-jones-avtcore-private-media-reqts]
Jones, P. et al., "Requirements for Private Media in a Switched Conferencing Environment", Work in Progress, March 2015.

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

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

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

13. Acknowledgments

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

Authors' Addresses

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

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

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

Email: nermeen@cisco.com

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

Email: dbenham@cisco.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: September 7, 2015

P. Jones (Ed.)
N. Ismail
D. Benham
N. Buckles
Cisco Systems
J. Mattsson
Y. Cheng
Ericsson
R. Barnes
Mozilla
March 7, 2015

Requirements for Private Media in a Switched Conferencing Environment
draft-jones-avtcore-private-media-reqts-01

Abstract

This document specifies the requirements for ensuring the privacy and integrity of real-time media flows between two or more endpoints communicating in a switched conferencing environment. This document also provides a high-level overview of switched conferencing in order to establish a common understanding of the goals and objectives of this work.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Users of multimedia communication products and services have privacy expectations that are largely satisfied with the use of SRTP [RFC3711] and related technologies when communicating point-to-point over the Internet. When communicating in a conferencing environment with two or more participants, though, it is necessary for an endpoint to share the SRTP master key and salt with the conference server so that it can authenticate and decrypt received RTP and RTCP packets. The conference server also needs the master key and salt in order to transmit media packets it receives to other participants in the conference. The need for conferencing servers to have the master key is a security risk for users.

Within a corporate or other isolated environment where conferencing servers are tightly controlled, this security risk can be effectively managed. However, managing this risk is becoming increasing difficult as conferencing resources are being deployed in networks that are less trusted, including virtualized conferencing servers deployed in cloud environments.

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

It is possible to ensure communication privacy within the context of a switched conferencing environment with limited changes in the security mechanisms used today. This document discusses this possibility in more detail and presents a set of requirements for meeting this objective.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Terminology

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

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

Switching conference server - A conference server that does not decrypt RTP media flows or perform processing on the media payload, but instead simply forwards the received media from a sender to the other participants in a multimedia conference. A switching conference server may modify some RTP headers.

4. Background

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

```
                   +--------------------+
+---+ --{A}-->     |                    | <--{C}-- +---+
| A |              | Media Composition  |          | C |
+---+ <-{BCD}-     |                    | -{ABD}-> +---+
                   |    Transcoders     |
+---+ --{B}-->     |    Transraters     | <--{D}-- +---+
| B |              |                    |          | D |
+---+ <-{ACD}-     |  Decrypt/Encrypt   | -{ABC}-> +---+
                   +--------------------+
```

Figure 1 - Traditional Conferencing Server

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

To enable the traditional conferencing server to perform its job, the server establishes an SRTP session with each of the conference participants so that it can get the keys required to decrypt and encrypt media flows from and to each participant. This means that the conference server is necessarily a fully trusted entity in the communication path. Anytime these servers are deployed in a network that is not tightly controlled, it increases the risk that an attacker might gain access to cryptographic key material, thus allowing the attacker to be able to see and listen to ongoing conferences. In some instances, depending on how the hardware is designed and how keys and certificates are managed, it might be possible for an attacker to see and listen to previously recorded conferences or future conferences.

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

5. Motivation for Private Media in Switched Conferencing

5.1. Switched Conferencing in Cloud Services

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

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

With a switching conference server, each participant transmits media to the server as it would with a traditional conferencing server. However, the switching conference server merely forwards media to the other participants in the conference (where the other participant may be associated with a cascaded conference server or an endpoint on the same server), leaving composition to the receiving endpoint. Since some endpoints may have a limited amount of bandwidth, each endpoint might negotiate with the switching conference server to receive only a subset of the available media flows. Each transmitting endpoint might also send multiple media flows of varying frame sizes and/or frame rates (e.g., simulcast or scalability layers), so that the server can select the streams most appropriate for each receiver’s bandwidth and capabilities. This allows, for example, an endpoint to receive and display higher quality video for the active speaker and thumbnails for other participants. It is also worth noting that, for switched media to work successfully, each endpoint in the conference must support the media formats transmitted by all other entities in the conference. More modern endpoints support multiple codecs and formats, making this commercially practical.

Figure 2 depicts an example of a switching conference server wherein each participant is receiving the media flows transmitted by each of the other participants in the conference.

```
                     +---------------------+
         +---+ --{A}--> |                     | <-{C}--- +---+
         | A | <-{B}--- |Switching Conference| --{A}--> | C |
         |   | <-{C}--- |       Server        | --{B}--> |   |
         +---+ <-{D}--- |                     | --{D}--> +---+
                        |       Packet        |
         +---+ --{B}--> |   Authentication    | <-{D}--- +---+
         | B | <-{A}--- |                     | --{A}--> | D |
         |   | <-{C}--- |                     | --{B}--> |   |
         +---+ <-{D}--- |    Media Privacy    | --{C}--> +---+
                        +---------------------+
```

Figure 2 - Switching Conference Server

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

By using methods such as those described in [RFC6464], it is possible for the switching conference server to transmit the appropriate audio and video flows to conference participants without having knowledge of the contents of the encrypted media. The examples that follow help to illustrate this point.

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

```
                         +--------------------+
            +---+ --{A}--> |                    | <--{C}-- +---+
            | A |          |Switching Conference|          | C |*
            +---+ <-{C}--- |       Server       | ---{A}-> +---+
                           |                    |
            +---+ --{B}--> |                    | <--{D}-- +---+
            | B |          |                    |          | D |
            +---+ <-{C}--- |                    | ---{C}-> +---+
                           +--------------------+
```

Figure 3 - Endpoint "C" is the Active Speaker

```
                           +--------------------+
            +---+ --{A}--> |                    | <--{C}-- +---+
            | A |          |Switching Conference|          | C |
            +---+ <-{B}--- |       Server       | ---{B}-> +---+
                           |                    |
            +---+ --{B}--> |                    | <--{D}-- +---+
           *| B |          |                    |          | D |
            +---+ <-{C}--- |                    | ---{B}-> +---+
                           +--------------------+
```

Figure 4 - Endpoint "B" is the Active Speaker

```
                           +--------------------+
            +---+ --{A}--> |                    | <--{C}-- +---+
           *| A |          |Switching Conference|          | C |
            +---+ <-{B}--- |       Server       | ---{A}-> +---+
                           |                    |
            +---+ --{B}--> |                    | <--{D}-- +---+
            | B |          |                    |          | D |
            +---+ <-{A}--- |                    | ---{A}-> +---+
                           +--------------------+
```

Figure 5 - Endpoint "A" is the Active Speaker

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

5.2. Private Media Security through Switching

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

The switching conference server just forwards media received to each of the other participants in the conference.

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

Thus, there is a need to utilize an end-to-end encryption and authentication key (or pair of keys) and a hop-by-hop encryption and authentication key (or pair of keys). The purpose for the hop-by-hop encryption key is to optionally encrypt RTP header extensions. The current SRTP specification and related specifications do not define use of a dual-key approach presently. However, such an approach is possible and would result in ensuring the privacy of media while also enabling the more scalable switched conferencing model.

The assumption is that no changes are made to SRTCP, i.e. SRTCP is protected hop-by-hop with a single security context.

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

6. Private Media Trust Model

The architecture suggested in this specification enables switching conference servers to be hosted in domains in which the network elements may have low trust, or where the trustworthiness is uncertain. This does not mean that the service provider is untrusted; it simply means that high trust is not required. This has the benefit of protecting the endpoints in the case of external attacks against the conference server.

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

There are very few elements that need to be trusted. However, it is also recognized that in certain deployment models, some elements that are classified as untrusted might be placed into the trusted domain and considered trusted. This specification is not intended to prevent such deployment models, but it does not rely upon them.

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

```
                                     |
                                     |
                       +--------------------------------------------+
                       v             |                              |
                 +----------+        |       +-----------------+    |
                 | Endpoint |--------------> | Call Processing |    |
                 +----------+        |       +-----------------+    |
                    ^                |         ^     ^              |
      Trusted       |                |         |     |       +------+
      Elements      |                |         |     |       |
                    |  +-----------------------+     |       |
                    |  |             |               v       v
                    |  |             |     +----------------------+
                    |  |  +--------------> | Switching Conference |
                    |  |  |          |     |       Server         |
                    v  v  v          |     +----------------------+
              +----------------+     |
              | Key Management |     |            Untrusted
              |    Function    |     |            Elements
              +----------------+     |
                                     |
                                     |
```

Figure 6 - Relationship of Trusted and Untrusted Elements

6.1. Trusted Elements

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

The other trusted element is a key management function (KMF). This function is responsible for providing cryptographic keys to the endpoints for encrypting and authenticating media content. The KMF is also responsible for providing cryptographic keys to the conferencing resources to enable authentication of media packets received by a conference participant. Interaction between the KMF and untrusted call processing functions may be necessary to ensure conference participants are delivered the appropriate keys or are directed to the appropriate conference server. It is expected that the KMF will be tightly controlled and managed to prevent exploitation by an adversary, as any kind of security compromise of the KMF puts the security of all conferences at risk.

6.2. Untrusted Elements

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

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

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

The final element is the switching conference server, which is responsible for forwarding encrypted media packets and conference control information to endpoints in the conference. It is also responsible for conveying secured signaling between the endpoints and the key management function, acquiring per-hop authentication keys from the KMF, and performing per-hop authentication operations for media packets. This function might also aggregate conference control information and initiate various conference control requests. Forwarding of media packets requires that the switching conference server have access to RTP headers or header extensions and potentially modify those message elements, but the actual media content MUST not be decipherable by the switching conference server.

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

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

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

7. Goals and Non-Goals

7.1. Goals

7.1.1. Ensure End-To-End Confidentiality

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

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

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

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

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

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

A switched conferencing model would yield the efficiencies desired.

7.1.4. Support Cloud-Based Conferencing

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

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

7.1.5. Limiting a User's Access to Content

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

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

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

7.1.6. Compatibility with the WebRTC Security Architecture

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

The browser is reliant on an external calling service to convey signaling information that may open the door for a man-in-the-middle attack, such as the conveyance of certificate fingerprints over the interface between the browser and the calling service. However, as described in [I.D-rtcweb-security-arch], the browser may utilize additional services, such as a trusted identify provider, to mitigate such risks.

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

7.2. Non-Goals

7.2.1. Securing the Endpoints

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

7.2.2. Concealing that Communication Occurs

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

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

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

7.2.3. Individual Media Source Authentication

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

There exist solutions that can provide individual media source authentication (e.g., TESLA). However, they impact the performance or security properties they provide. Thus, further study is required to determine impact and resulting security properties if desired to have individual source authentication.

7.2.4. Support for Multicast in Switched Conferencing

Multicast traffic is, by design, transmitted to every participant in a conference. The focus of this document is only on centralized unicast conferencing that utilizes a switched conferencing architecture.

8. Requirements

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

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

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

PM-02: Solution MUST maintain all current SRTP security goals, namely the ability to provide for end-to-end confidentiality, provide for hop-by-hop replay protection, and ensure hop-by-hop and end-to-end message integrity. {Editor's Note: Question asked, "Does this include third parties?" Jonathan Lennox to suggest ways to make this more concrete.}

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

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

PM-05: The switching conference server MUST be capable of making changes to the RTP header and, optionally, the RTP header extensions.

PM-06: The SRTP cryptographic context, which is identified in part by an SSRC, contains transform-independent parameters used by the sending endpoint, including the RTP packet sequence number and rollover counter (ROC), required for packet decryption and authentication that, along with the value of the SSRC, MUST be protected end-to-end.

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

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

PM-09: It MUST be possible for the switching conference server to determine if a received media packet was transmitted by a conference participant in possession of the end-to-end media encryption keys and hop-by-hop authentication keys.

PM-10: It MUST be possible for a conference to be optionally re-keyed as desired, such as each time a participant joins or leaves the conference. {Editor's note: Who is allowed to know who leaves and joins? Do you trust the conference server to tell you reliably?}

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

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

9. IANA Considerations

There are no IANA considerations for this document.

10. Security Considerations

[TBD]

11. References

11.1. Normative References

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

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

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

[I.D-rtcweb-security-arch] E. Rescorla, "WebRTC Security Architecture", Work in Progress, July 2014.

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

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

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

11.2. Informative References

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

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

12. Acknowledgments

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

13. Contributors

[TBD]

Authors' Addresses

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

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

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

Email: nermeen@cisco.com

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

Email: dbenham@cisco.com

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

Email: nbuckles@cisco.com

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

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

Yi Cheng
Ericsson
SE-164 80 Stockholm

Sweden

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

Richard Barnes
Mozilla
331 E Evelyn Ave.
Mountain View
USA

Email: rlb@ipv.sx

AVTCORE M. Petit-Huguenin
Internet-Draft Impedance Mismatch
Updates: 5764 (if approved) G. Salgueiro
Intended status: Standards Track Cisco Systems
Expires: September 7, 2015 March 6, 2015

Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)
draft-petithuguenin-avtcore-rfc5764-mux-fixes-02

Abstract

This document defines how Datagram Transport Layer Security (DTLS), Real-time Transport Protocol (RTP), Real-time Transport Control Protocol (RTCP), Session Traversal Utilities for NAT (STUN), and Traversal Using Relays around NAT (TURN) packets are multiplexed on a single receiving socket. It overrides the guidance from SRTP Extension for DTLS [RFC5764], which suffered from three issues described and fixed in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Section 5.1.2 of Secure Real-time Transport Protocol (SRTP) Extension for DTLS [RFC5764] defines a scheme for a Real-time Transport Protocol (RTP) [RFC3550] receiver to demultiplex Datagram Transport Layer Security (DTLS) [RFC6347], Session Traversal Utilities for NAT (STUN) [RFC5389] and Secure Real-time Transport Protocol (SRTP)/Secure Real-time Transport Control Protocol (SRTCP) [RFC3711] packets that are arriving on the RTP port. Unfortunately, this demultiplexing scheme has created three problematic issues:

1. It implicitly allocated codepoints for new STUN methods without an IANA registry reflecting these new allocations.

2. It implicitly allocated codepoints for new Transport Layer Security (TLS) ContentTypes without an IANA registry reflecting these new allocations.

3. It did not take into account the fact that the Traversal Using Relays around NAT (TURN) usage of STUN can create TURN channels that also need to be demultiplexed with the other packet types explicitly mentioned in Section 5.1.2 of RFC 5764.

These flaws in the demultiplexing scheme were unavoidably inherited by other documents, such as [RFC7345] and [I-D.ietf-mmusic-sdp-bundle-negotiation]. These will need to be corrected with the updates this document provides when it become normative.

1.1. Implicit Allocation of Codepoints for New STUN Methods

The demultiplexing scheme in [RFC5764] states that the receiver can identify the packet type by looking at the first byte. If the value of this first byte is 0 or 1, the packet is identified to be STUN. The problem that arises as a result of this implicit allocation is that this restricts the codepoints for STUN methods (as described in Section 18.1 of [RFC5389]) to values between 0x000 and 0x07F, which in turn reduces the number of possible STUN method codepoints assigned by IETF Review (i.e., the range from (0x000 - 0x7FF) from 2048 to only 128 and entirely obliterating those STUN method codepoints assigned by Designated Expert (i.e., the range 0x800 - 0xFFF). In fact, RFC 5764 implicitly (and needlessly) allocated a very large range of STUN methods, but at a minimum the IANA STUN Methods registry should properly reflect this.

There are only a few STUN method codepoints currently allocated. For this reason, simply marking the implicit allocations made by RFC 5764 in the STUN Method registry may create a shortage of codepoints at a time when interest in STUN and STUN Usages (especially TURN) is growing rapidly. Consequently, this document also changes the RFC 5764 packet identification algorithm to expand the range assigned to the STUN protocol from 0 - 1 to 0 - 19, as the values 2-19 are unused.

In addition to explicitly allocating STUN methods codepoints from 0x500 to 0xFFF as Reserved values, this document also updates the IANA registry such that the STUN method codepoints assigned via IETF Review are in the 0x000-0x27F range and those assigned via Designated Expert are in the 0x280-0x4FF range. The proposed changes to the STUN Method Registry is:

OLD:

```
0x000-0x7FF     IETF Review
0x800-0xFFF     Designated Expert
```

NEW:

```
0x000-0x27F     IETF Review
0x280-0x4FF     Designated Expert
0x500-0xFFF     Reserved
```

1.2. Implicit Allocation of New Codepoints for TLS ContentTypes

The demultiplexing scheme in [RFC5764] dictates that if the value of the first byte is between 20 and 63 (inclusive), then the packet is identified to be DTLS. The problem that arises is that this restricts the TLS ContentType codepoints (as defined in Section 12 of [RFC5246]) to this range, and by extension implicitly allocates ContentType codepoints 0 to 19 and 64 to 255. Unlike STUN, TLS is a mature protocol that is already well established and widely implemented and thus we expect only relatively few new codepoints to be assigned in the future. With respect to TLS packet identification, this document simply explicitly reserves the codepoints from 0 to 19 and from 64 to 255 so they are not inadvertently assigned in the future.

1.3. Multiplexing of TURN Channels

When used with ICE [RFC5245], an RFC 5764 implementation can receive packets on the same socket from three different paths, as shown in Figure 1:

1. Directly from the source

2. Through a NAT

3. Relayed by a TURN server

```
       +------+
       | TURN |<------------------------+
       +------+                         |
          |                             |
          | +-------------------------+ |
          | |                         | |
          v v                         | |
   NAT -----------                    | |
          | | +---------------------+ | |
          | | |                     | | |
          v v v                     | | |
      +----------+              +----------+
      | RFC 5764 |              | RFC 5764 |
      +----------+              +----------+
```

Figure 1: Packet Reception by an RFC 5764 Implementation

Even if the ICE algorithm succeeded in selecting a non-relayed path, it is still possible to receive data from the TURN server. For instance, when ICE is used with aggressive nomination the media path can quickly change until it stabilizes. Also, freeing ICE candidates is optional, so the TURN server can restart forwarding STUN connectivity checks during an ICE restart.

TURN channels are an optimization where data packets are exchanged with a 4-byte prefix, instead of the standard 36-byte STUN overhead (see Section 2.5 of [RFC5766]). The problem is that the RFC 5764 demultiplexing scheme does not define what to do with packets received over a TURN channel since these packets will start with a first byte whose value will be between 64 and 127 (inclusive). If the TURN server was instructed to send data over a TURN channel, then the current RFC 5764 demultiplexing scheme will reject these packets. Current implementations violate RFC 5764 for values 64 to 127 (inclusive) and they instead parse packets with such values as TURN. In order to prevent future documents from assigning values from the unused range to a new protocol, this document modifies the RFC 5764 demultiplexing algorithm to properly account for TURN channels.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "must" or "Must"), they have their usual English meanings, and are not to be interpreted as RFC 2119 key words.

3. RFC 5764 Updates

This document updates the text in Section 5.1.2 of [RFC5764] as follows:

OLD TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is 0 or 1, then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. This process is summarized in Figure 3.

```
             +----------------+
             | 127 < B < 192 -+--> forward to RTP
             |                |
 packet -->  |  19 < B < 64  -+--> forward to DTLS
             |                |
             |       B < 2   -+--> forward to STUN
             +----------------+
```

Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm. Here the field B denotes the leading byte of the packet.

END OLD TEXT

NEW TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is in between 0 and 19 (inclusive), then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. If the value is between 64 and 127 (inclusive), the packet is TURN Channel. This process is summarized in Figure 3.

```
                    +----------------+
                    | 127 < B < 192 -+--> forward to RTP
                    |                |
                    |  63 < B < 128 -+--> forward to TURN Channel
        packet -->  |                |
                    |  19 < B < 64  -+--> forward to DTLS
                    |                |
                    |       B < 20  -+--> forward to STUN
                    +----------------+
```

Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm. Here the field B denotes the leading byte of the packet.

END NEW TEXT

[[Note: we may want to use "<=" instead of "<" to make it easier on implementers.]]

4. Implementation Status

[[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note that there is currently no implementation declared in this section, but the intent is to add RFC 6982 templates here from implementers that support the modifications in this document.

5. Security Considerations

This document simply updates existing IANA registries and does not introduce any specific security considerations beyond those detailed in [RFC5764].

6. IANA Considerations

6.1. STUN Methods

This specification contains the registration information for 2816 STUN Methods codepoints, as explained in Section 1.1 and in accordance with the procedures defined in Section 18.1 of [RFC5389].

Value: 0x500-0xFFF

Name: Reserved

Reference: RFC5764, RFCXXXX

This specification also reassigns the ranges in the STUN Methods Registry as follow:

Range: 0x000-0x27F

Registration Procedures: IETF Review

Range: 0x280-0x4FF

Registration Procedures: Designated Expert

6.2. TLS ContentType

This specification contains the registration information for 212 TLS ContentType codepoints, as explained in Section 1.2 and in accordance with the procedures defined in Section 12 of [RFC5246].

Value: 0-19

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

Value: 64-255

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

6.3. TURN Channel Numbers

This specification contains the registration information for 32768 TURN Channel Numbers codepoints, as explained in Section 1.3 and in accordance with the procedures defined in Section 18 of [RFC5766].

Value: 0x8000-0xFFFF

Name: Reserved

Reference: RFCXXXX

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

7. Acknowledgements

The implicit STUN Method codepoint allocations problem was first reported by Martin Thomson in the RTCWEB mailing-list and discussed further with Magnus Westerlund.

Thanks to Simon Perreault, Colton Shields and Cullen Jennings for the comments, suggestions, and questions that helped improve this document.

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.

[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.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

8.2. Informative References

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[RFC7345] Holmberg, C., Sedlacek, I., and G. Salgueiro, "UDP Transport Layer (UDPTL) over Datagram Transport Layer Security (DTLS)", RFC 7345, August 2014.

[I-D.ietf-mmusic-sdp-bundle-negotiation] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-12 (work in progress), October 2014.

Appendix A. Release notes

This section must be removed before publication as an RFC.

A.1. Modifications between draft-petithuguenin-avtcore-rfc5764-mux-fixes-00 and draft-petithuguenin-avtcore-rfc5764-mux-fixes-01

- Change affiliation.

Authors' Addresses

Marc Petit-Huguenin
Impedance Mismatch

Email: marc@petit-huguenin.org

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com