Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 10 April 2026

P. Thatcher
Microsoft
Y. Fablet
Apple

7 October 2025

RTP Payload Format for SFrame
draft-ietf-avtcore-rtp-sframe-01

Abstract

This document describes the RTP payload format of SFrame.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-wg-avtcore.github.io/draft-ietf-avtcore-rtp-sframe/draft-ietf-avtcore-rtp-sframe.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-avtcore-rtp-sframe/.

Discussion of this document takes place on the Audio/Video Transport Core Maintenance Working Group mailing list (mailto:avt@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/avt/. Subscribe at https://www.ietf.org/mailman/listinfo/avt/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-avtcore/draft-ietf-avtcore-rtp-sframe.

Status of This Memo

Table of Contents

1. Introduction

SFrame [RFC9605] describes an end-to-end encryption and authentication mechanism for media data in a multiparty conference call, in which central media servers (SFUs) can access the media metadata needed to make forwarding decisions without having access to the actual media.

This document describes how to packetize a media frame encrypted using SFrame into RTP packets.

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. SFrame format

An SFrame ciphertext comprises a header and encrypted data. The SFrame header has a size varying between 1 to 17 bytes. The encrypted data can be of arbitrary length and is larger than the unencrypted data by a fixed overhead that depends on the encryption algorithm. The overhead can be up to 16 bytes.

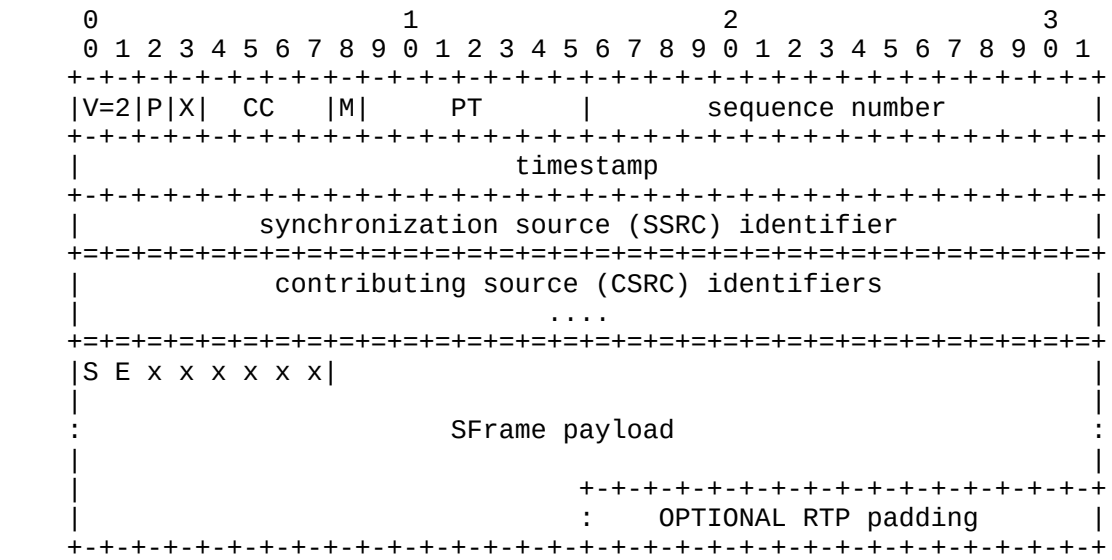
An SFrame ciphertext having an arbitrary long length, an application may decide to partition the data encrypted with SFrame small enough so that the SFrame ciphertext fits in a single RTP packet. We call this per-packet SFrame. This has the advantage of allowing to decrypt the content as soon as received.

An alternative is to encrypt the data, a media frame typically, and send the SFrame ciphertext over several RTP packets. We call this per-frame SFrame. This has the advantage of limiting the SFrame overhead, especially for video frames. This alternative is also compatible with [WebRTC_Encoded_Transform], which is important for backward compatibility of existing services.

The RTP format presented in this document supports both alternatives.

4. RTP Header Usage

The general RTP payload format for SFrame is depicted below.



The first byte of the RTP payload is the SFrame RTP header.

The S bit of the SFrame RTP header MUST be 0 for all fragments except for the first one of the SFrame frame.

The E bit of the SFrame RTP header MUST be 0 for all fragments except for the last one of the SFrame frame.

The 6 remaining bits of the SFrame RTP header are reserved for future use.

The payload type (PT) identifies the format of the media encrypted with SFrame.

The SSRC, timestamp, marker bit, and CSRCs of the SFrame RTP packets MUST be the same as those of the output of the media-format-specific packetization. The header extensions of the SFrame RTP packets SHOULD be the same as those of the output of the media-format-specific packetization, but some may be omitted if it is known that the omitted header extensions do not need to be duplicated on each SFrame RTP packet.

5. RTP Packetization of SFrame

SFrame packets can be generated either from RTP media packets or from media frames as defined by [WebRTC_Encoded_Transform].

For per-packet SFrame, the following processing is done, with a media frame as input:

1. Generate a group of RTP media packets from the media frame using a media-format-specific packetizer. The media-format-specific packetizer needs to be made aware of the SFrame overhead that happens to each RTP packet.

2. For each RTP packet of the group, encrypt its payload with SFrame.

3. Prepend to each RTP packet payload a SFrame RTP header with the S and E bits set to 1.

4. Send each RTP packet of the group.

For per-frame SFrame, the following processing is done, with a media frame as input:

1. Generate a SFrame ciphertext from the media frame data.

2. Fragment the SFrame ciphertext in a group of payloads so that RTP packets generated from them do not exceed the network maximum transmission unit size.

3. Prepend a zero byte as the SFrame RTP header to all payloads of the group.

4. Set the first bit S of the SFrame RTP header of the first packet to 1.

5. Set the second bit E of the SFrame RTP header of the last packet to 1.

6. Generate a group of RTP packets from the group of payloads, using the media frame to generate the RTP header, including RTP header extensions.

7. Send each RTP packet of the RTP packet group.

6. RTP depacketization of SFrame

Reception of SFrame packets is done as follows:

1. The fragments of a given SFrame ciphertext are grouped together in order of the RTP sequence number, the first packet of the group having its S bit set to 1 and the last packet of the group havint its E bit set to 1. All packets in between the first and last needs to be in the group.

2. Concatenate the payloads of all packets of the group to form the SFrame ciphertext.

3. Decrypt the SFrame ciphertext to obtain the media decrypted data.

4. If per-packet SFrame is being used, the following processing is done:

    1. assert that the group of packets consist of a single packet.

    2. Set the media decrypted data as the payload of the packet and send the packet to the media-format-specific RTP depacketizer.

    3. If the depacketizer cannot generate a media frame yet, abort these steps. Otherwise, generate a media frame from the depacketizer.

5. If per-frame SFrame is being used, the following processing is done:

    1. assert that the group of packets all have the same payload type.

    2. Extract the media metadata from the group of packets.

    3. Generate a media frame from the media decrypted data and the media metadata.

6. Send the media frame to the receiving pipeline.

7. SFrame SDP negotiation

SFrame packetization is indicated via a new "a=sframe" SDP attribute defined in this specification. This attribute is used at media level, it does not appear at session level.

The presence of the "a=sframe" attribute in a media section (in either an offer or an answer) indicates that the endpoint is expecting to receive RTP packets encrypted with SFrame for that media section, as defined below.

Once each peer has verified that the other party expects to receive SFrame RTP packets, senders are expected to send SFrame encrypted RTP packets. If one peer expects to use SFrame for a media section and identifies that the other peer does not support it, the peer is expected to stop the transceiver associated to the media section, which will generate a zero port for that m-section.

When SFrame is in use for that media section, it will apply to the relevant media encodings defined for that media section. This includes RTP payload types bound to media packetizers and media depacketizers as defined in [RFC7656], typically audio formats such as Opus and RTP video formats such as H264. This notably includes RTP payload types representing [WebRTC_Encoded_Transform] encoded video frame formats (https://w3c.github.io/webrtc-encoded-transform/#dom-rtcencodedvideoframe-data) and encoded audio frame formats (https://w3c.github.io/webrtc-encoded-transform/#dom-rtcencodedaudioframe-data).

This does not include RTP-Based Redundancy mechanisms as defined in {!RFC7656}}. For instance, RTX defined in [RFC4588] will retransmit SFrame based packets. Forward error correction formats as defined in [RFC5109] will protect the encrypted content. For Redundant Audio Data, known as RED, as defined in [RFC2198], a RED packetizer will take as input SFrame encrypted media data instead of unencrypted media data.

If BUNDLE is in use and the "a=sframe" attribute is present for a media section but not for another media section of the same BUNDLE, payload types for media encodings that are relevant for SFrame MUST not be reused between the two media sections.

Questions:

1. Should we precise how RTX/FEC works with SFrame packetization? No impact AFAIK since RTX/FEC would work on packets (whether SFramed or not).

2. Is RED current proposal (transmit SFrame ciphertext blocks) good enough? An alternative is to have SFrame being applied on the entire RED packet payload.

3. Should we allow a=sframe at session level to mean that all media sections want sframe?

Here is an example of SFrame being negotiated for audio (opus and CN) and for video (H264 and VP8):

```
m=audio 50000 RTP/SAVPF 10 11
a=sframe
a=rtpmap:10 opus/48000/2
a=rtpmap:11 CN/8000

m=video 50002 RTP/SAVPF 100 101
a=sframe
a=rtpmap:100 H264/90000
a=rtpmap:101 VP8/90000
```

8. Security Considerations

This document is subject to the security considerations of SFrame.

9. IANA Considerations

None

10. Normative References

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

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J.C., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, DOI 10.17487/RFC2198, September 1997, <https://www.rfc-editor.org/rfc/rfc2198>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/rfc/rfc4588>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <https://www.rfc-editor.org/rfc/rfc5109>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/rfc/rfc7656>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9605] Omara, E., Uberti, J., Murillo, S. G., Barnes, R., Ed., and Y. Fablet, "Secure Frame (SFrame): Lightweight Authenticated Encryption for Real-Time Media", RFC 9605, DOI 10.17487/RFC9605, August 2024, <https://www.rfc-editor.org/rfc/rfc9605>.

[WebRTC_Encoded_Transform] World Wide Web Consortium, "WebRTC Encoded Transform", May 2025, <https://w3c.github.io/webrtc-encoded-transform/>.

Authors' Addresses

Peter Thatcher
Microsoft
Email: pthatcher@microsoft.com

Youenn Fablet
Apple
Email: youenn@apple.com