AVTCore
Internet-Draft
Obsoletes: RFC5285 (if approved)
Intended status: Standards Track
Expires: May 1, 2017

R. Even, Ed.
Huawei Technologies
D. Singer
Apple, Inc.
H. Desineni

October 28, 2016

A General Mechanism for RTP Header Extensions
draft-ietf-avtcore-rfc5285-bis-04.txt

Abstract

This document provides a general mechanism to use the header extension feature of RTP (the Real-Time Transport Protocol). It provides the option to use a small number of small extensions in each RTP packet, where the universe of possible extensions is large and registration is de-centralized. The actual extensions in use in a session are signaled in the setup information for that session. The document obsoletes RFC5285

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on May 1, 2017.

Table of Contents

1. Introduction

The RTP specification [RFC3550] provides a capability to extend the RTP header. It defines the header extension format and rules for its use in Section 5.3.1. The existing header extension method permits at most one extension per RTP packet, identified by a 16-bit identifier and a 16-bit length field specifying the length of the header extension in 32-bit words.

This mechanism has two conspicuous drawbacks. First, it permits only one header extension in a single RTP packet. Second, the specification gives no guidance as to how the 16-bit header extension identifiers are allocated to avoid collisions.

This specification removes the first drawback by defining a backward-compatible and extensible means to carry multiple header extension elements in a single RTP packet. It removes the second drawback by defining that these extension elements are named by URIs, defining an IANA registry for extension elements defined in IETF specifications, and a Session Description Protocol (SDP) method for mapping between the naming URIs and the identifier values carried in the RTP packets.

This header extension applies to RTP/AVP (the Audio/Visual Profile) and its extensions.

This document obsoletes [RFC5285] and removes a limitation from RFC5285 that did not allow sending both one byte and two bytes header extensions in the same RTP stream

2. Requirements Notation

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].

3. Design Goals

The goal of this design is to provide a simple mechanism whereby multiple identified extensions can be used in RTP packets, without the need for formal registration of those extensions but nonetheless avoiding collision.

This mechanism provides an alternative to the practice of burying associated metadata into the media format bit stream. This has often been done in media data sent over fixed-bandwidth channels. Once this is done, a decoder for the specific media format needs to extract the metadata. Also, depending on the media format, the metadata can be added at the time of encoding the media so that the bit-rate used for the metadata is taken into account. But the metadata can be unknown at that time. Inserting metadata at a later time can cause a decode and re-encode to meet bit-rate requirements.

In some cases, a more appropriate, higher-level mechanism can be available, and if so, it can be used. For cases where a higher-level mechanism is not available, it is better to provide a mechanism at the RTP level than have the metadata be tied to a specific form of media data.

4. Packet Design

4.1. General

The following design is fit into the "header extension" of the RTP extension, as described above.

The presence and format of this header extension and its contents are negotiated or defined out-of-band, such as through signaling (see below for SDP signaling). The value defined for an RTP extension (defined below for the one-byte and two-byte header forms) is only an architectural constant (e.g., for use by network analyzers); it is the negotiation/definition (e.g., in SDP) that is the definitive indication that this header extension is present.

This specification updates the requirement from the RTP specification that the header extension "is designed so that the header extension MAY be ignored". To be specific, header extensions using this specification SHOULD be used for data that can safely be ignored by the recipient without affecting interoperability, there can be essential header extensions for interoperability and intermediaries SHOULD NOT remove such header extensions. Note that the support of header extension as specified in this recommendation is negotiated. RTP Header extensions MUST NOT be used when the presence of the extension has changed the form or nature of the rest of the packet in a way that is not compatible with the way the stream is signaled (e.g., as defined by the payload type). Valid examples might include metadata that is additional to the usual RTP information, e.g. Audio level from Client to mixer [RFC6464].

The RTP header extension is formed as a sequence of extension elements, with possible padding. Each extension element has a local identifier and a length. The local identifiers MAY be mapped to a larger namespace in the negotiation (e.g., session signaling).

4.1.1. transmission considertions

As is good network practice, data SHOULD only be transmitted when needed. The RTP header extension SHOULD only be present in a packet if that packet also contains one or more extension elements, as defined here. An extension element SHOULD only be present in a packet when needed; the signaling setup of extension elements indicates only that those elements can be present in some packets, not that they are in fact present in all (or indeed, any) packets.

Some general considerations for getting the header extensions delivered to the receiver:

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely, what distribution would be needed to avoid getting all repetitions of the header extensions lost in a single burst.

2. If a set of packets are all needed to enable decoding, there is commonly no reason for including the header extension in all of these packets, as they share fate. Instead, at most one instance of the header extension per independently decodable set of media data would be a more efficient use of the bandwidth.

3. How early the Header Extension item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.

4. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

As a summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be optimal. N is selected so that the header extension target delivery probability reaches 1-P^N, where P is the probability of packet loss. For point to point or small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has reached all receivers and stop further repetitions. Feedback that can be used includes the RTCP XR Loss RLE report block [RFC3611], which will indicate successful delivery of particular packets. If the RTP/AVPF Transport Layer Feedback Messages for generic NACK [RFC4585] is used, it can indicate the failure to deliver an RTP packet with the header extension, thus indicating the need for further repetitions. The normal RTCP report blocks can also provide an indicator of successful delivery, if no losses are indicated for a reporting interval covering the RTP packets with the header extension. Note that loss of an RTCP packet reporting on an interval where RTP header extension packets were sent, does not necessarily mean that the RTP header extension packets themselves were lost.

4.1.2. Header Extension type consideration

Each extension element in a packet has a local identifier (ID) and a length. The local identifiers present in the stream MUST have been negotiated or defined out-of-band. There are no static allocations of local identifiers. Each distinct extension MUST have a unique ID. The value 0 is reserved for padding and MUST NOT be used as a local identifier.

There are two variants of the extension: one-byte and two-byte headers. Since it is expected that (a) the number of extensions in any given RTP session is small and (b) the extensions themselves are small, the one-byte header form is preferred and MUST be supported by all receivers. A stream MUST contain only one-byte or two-byte headers unless it is known that all recipients support mixing, either by offer/answer negotiation (see section 6) or by out-of-band knowledge. Each RTP packet with an RTP header extension following this specification will indicate if it contains one or two byte header extensions through the use of the "defined by profile" field. Only the extension element types that match the header extension format, i.e. one- or two-byte, MUST be used in that RTP packet. Transmitters SHOULD NOT use the two-byte form when all extensions are small enough for the one-byte header form. Transmitters that intend to send the two-byte form SHOULD use IDs above 14 if they want to let the Receivers know that they intend to use two-byte form, for example if the RTP header extension is longer than 16 bytes. A transmitter MAY be aware that an intermediary may add RTP header extensions in this case, the transmitter SHOULD use two-byte form.

A sequence of extension elements, possibly with padding, forms the header extension defined in the RTP specification. There are as many extension elements as fit into the length as indicated in the RTP header extension length. Since this length is signaled in full 32-bit words, padding bytes are used to pad to a 32-bit boundary. The entire extension is parsed byte-by-byte to find each extension element (no alignment is needed), and parsing stops at the earlier of the end of the entire header extension, or in one-byte headers only case, on encountering an identifier with the reserved value of 15.

In both forms, padding bytes have the value of 0 (zero). They MAY be placed between extension elements, if desired for alignment, or after the last extension element, if needed for padding. A padding byte does not supply the ID of an element, nor the length field. When a padding byte is found, it is ignored and the parser moves on to interpreting the next byte.

Note carefully that the one-byte header form allows for data lengths between 1 and 16 bytes, by adding 1 to the signaled length value

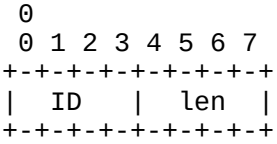
(thus, 0 in the length field indicates 1 byte of data follows). This allows for the important case of 16-byte payloads. This addition is not performed for the two-byte headers, where the length field signals data lengths between 0 and 255 bytes.

Use of RTP header extensions will reduce the efficiency of RTP header compression, since the header extension will be sent uncompressed unless the RTP header compression module is updated to recognize the extension header. If header extensions are present in some packets, but not in others, this can also reduce compression efficiency by requiring an update to the fixed header to be conveyed when header extensions start or stop being sent. The interactions of the RTP header extension and header compression is explored further in [RFC2508] and [RFC3095].

4.2. One-Byte Header

In the one-byte header form of extensions, the 16-bit value REQUIRED by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", MUST have the fixed bit pattern 0xBEDE (the first version of this specification was written on the feast day of the Venerable Bede).

Each extension element MUST starts with a byte containing an ID and a length:



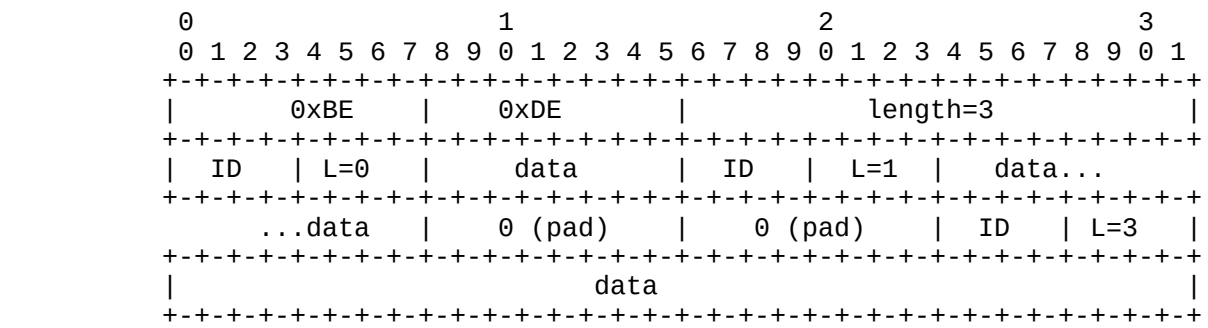
The 4-bit ID is the local identifier of this element in the range 1-14 inclusive. In the signaling section, this is referred to as the valid range.

The local identifier value 15 is reserved for future extension and MUST NOT be used as an identifier. If the ID value 15 is encountered, its length field MUST be ignored, processing of the entire extension MUST terminate at that point, and only the extension elements present prior to the element with ID 15 SHOULD be considered.

The 4-bit length is the number minus one of data bytes of this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows, and a value of 15 (the maximum) indicates element data of 16 bytes.

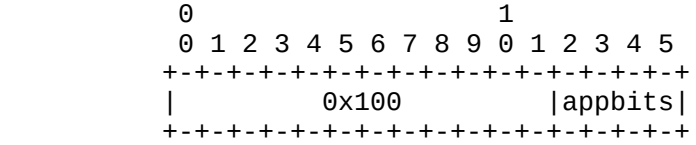
(This permits carriage of 16-byte values, which is a common length of labels and identifiers, while losing the possibility of zero-length values -- which would often be padded anyway.)

An example header extension, with three extension elements, and some padding follows:



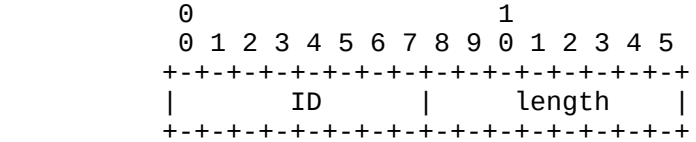
4.3. Two-Byte Header

In the two-byte header form, the 16-bit value defined by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", is defined as shown below.



The appbits field is 4 bits that are application-dependent and MAY be defined to be any value or meaning, and are outside the scope of this specification. For the purposes of signaling, this field is treated as a special extension value assigned to the local identifier 256. If no extension has been specified through configuration or signaling for this local identifier value 256, the appbits field SHOULD be set to all 0s by the sender and MUST be ignored by the receiver.

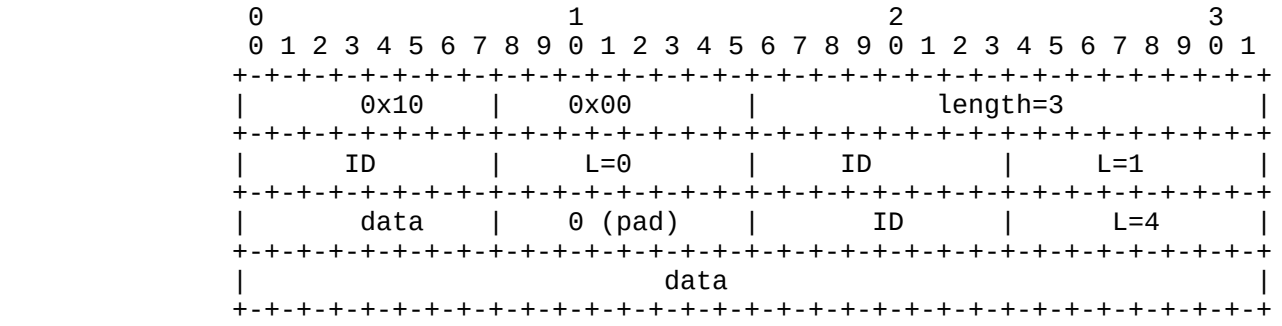
Each extension element starts with a byte containing an ID and a byte containing a length:



The 8-bit ID is the local identifier of this element in the range 1-255 inclusive. In the signaling section, the range 1-256 is referred to as the valid range, with the values 1-255 referring to extension elements, and the value 256 referring to the 4-bit field 'appbits' (above). Note that there is one ID space for both one-byte and two-byte form this means that the lower values (1-14) can be used in the 4-bit ID field in the one-byte header format as well.

The 8-bit length field is the length of extension data in bytes not including the ID and length fields. The value zero indicates there is no data following.

An example header extension, with three extension elements, and some padding follows:



5. SDP Signaling Design

The indication of the presence of this extension, and the mapping of local identifiers used in the header extension to a larger namespace, MUST be performed out-of-band, for example, as part of a SIP offer/answer exchange using SDP. This section defines such signaling in SDP.

A usable mapping MUST use IDs in the valid range, and each ID in this range MUST be used only once for each media (or only once if the mappings are session level). Mappings that do not conform to these rules MAY be presented, for instance, during offer/answer negotiation as described in the next section, but remapping to conformant values is necessary before they can be applied.

Each extension is named by a URI. That URI MUST be absolute, and precisely identifies the format and meaning of the extension. URIs that contain a domain name SHOULD also contain a month-date in the form mmyyyy. The definition of the element and assignment of the URI MUST have been authorized by the owner of the domain name on or very close to that date. (This avoids problems when domain names change

ownership.) If the resource or document defines several extensions, then the URI MUST identify the actual extension in use, e.g., using a fragment or query identifier (characters after a '#' or '?' in the URI).

Rationale: the use of URIs provides for a large, unallocated space, and gives documentation on the extension. The URIs do not have to be de-referencable, in order to permit confidential or experimental use, and to cover the case when extensions continue to be used after the organization that defined them ceases to exist.

An extension URI with the same attributes MUST NOT appear more than once applying to the same stream, i.e., at session level or in the declarations for a single stream at media level. (The same extension can, of course, be used for several streams, and can appear differently parameterized for the same stream.)

For extensions defined in RFCs, the URI used SHOULD be a URN starting "urn:ietf:params:rtp-hdrext:" and followed by a registered, descriptive name.

The registration requirements are detailed in the IANA Considerations section, below.

An example (this is only an example), where 'avt-example-metadata' is the hypothetical name of a header extension, might be:

urn:ietf:params:rtp-hdrext:avt-example-metadata

An example name not from the IETF (this is only an example) might be:

http://example.com/082005/ext.htm#example-metadata

The mapping MAY be provided per media stream (in the media-level section(s) of SDP, i.e., after an "m=" line) or globally for all streams (i.e., before the first "m=" line, at session level). The definitions MUST be either all session level or all media level; it is not permitted to mix the two styles. In addition, as noted above, the IDs used MUST be unique for each stream type for a given media, or for the session for session-level declarations.

Each local identifier potentially used in the stream is mapped to a string using an attribute of the form:

a=extmap:<value>["/"<direction>] <URI> <extensionattributes>

where <URI> is a URI, as above, <value> is the local identifier (ID) of this extension and is an integer in the valid range inclusive (0

is reserved for padding in both forms, and 15 is reserved in the one-byte header form, as noted above), and <direction> is one of "sendonly", "recvonly", "sendrecv", or "inactive" (without the quotes) with relation to the device being configured.

The formal BNF syntax is presented in a later section of this specification.

Example:

a=extmap:1 http://example.com/082005/ext.htm#ttime

a=extmap:2/sendrecv http://example.com/082005/ext.htm#xmeta short

When SDP signaling is used for the RTP session, it is the presence of the 'extmap' attribute(s) that is diagnostic that this style of header extensions is used, not the magic number indicated above.

6. SDP Signaling for support of mixed one byte and two bytes header extensions.

In order to allow for backward interoperability with systems that do not support mixing of one byte and two bytes header extensions this document defines the "a=extmap-allow-mixed" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the participant is capable of supporting this new mode. The attribute takes no value. This attribute can be used at the session or media levels. A participant that proposes the use of this mode SHALL itself support the reception of mixed one byte and two bytes header extensions.

The negotiation for mixed one byte and two bytes extension MUST be negotiated in offer/answer [RFC3264]. In the absence of negotiation using offer/answer, mixed headers MUST NOT occur unless the transmitter has some (out of band) knowledge that all potential recipients support this mode.

The formal definition of this attribute is:

Name: extmap-allow-mixed

Value:

Usage Level: session, media

Charset Dependent: no

Example:

a=extmap-allow-mixed

When doing SDP Offer/Answer [RFC3264] an offering client that wishes to use both one and two bytes extensions MUST include the attribute "a= extmap-allow-mixed " in the SDP offer. If "a= extmap-allow-mixed " is present in the offer SDP, the answerer that supports this mode and wishes to use it SHALL include the "a=extmap-allow-mixed " attribute in the answer. In cases the answer has been excluded, neither clients SHALL use mixed one bytes and two bytes extensions in the same RTP stream but MAY use one-byte or two-bytes form (see section 4.1.2).

7. Offer/Answer

The simple signaling described above for the extmap attribute MAY be enhanced in an offer/answer context, to permit:

- asymmetric behavior (extensions sent in only one direction),
- the offer of mutually exclusive alternatives, or
- the offer of more extensions than can be sent in a single session.

A direction attribute MAY be included in an extmap; without it, the direction implicitly inherits, of course, from the stream direction, or is "sendrecv" for session-level attributes or extensions of "inactive" streams. The direction MUST be one of "sendonly", "recvonly", "sendrecv", or "inactive" as specified in [RFC3264]

Extensions, with their directions, MAY be signaled for an "inactive" stream. It is an error to use an extension direction incompatible with the stream direction (e.g., a "sendonly" attribute for a "recvonly" stream).

If an offer or answer contains session-level mappings (and hence no media-level mappings), and different behavior is desired for each stream, then the entire set of extension map declarations MAY be moved into the media-level section(s) of the SDP. (Note that this specification does not permit mixing global and local declarations, to make identifier management easier.)

If an extension map is offered as "sendrecv", explicitly or implicitly, and asymmetric behavior is desired, the SDP MAY be modified to modify or add direction qualifiers for that extension.

If an extension is marked as "sendonly" and the answerer desires to receive it, the extension MUST be marked as "recvonly" in the SDP answer. An answerer that has no desire to receive the extension or

does not understand the extension SHOULD remove it from the SDP answer.

If an extension is marked as "recvonly" and the answerer desires to send it, the extension MUST be marked as "sendonly" in the SDP answer. An answerer that has no desire to, or is unable to, send the extension SHOULD remove it from the SDP answer.

Local identifiers in the valid range inclusive in an offer or answer MUST NOT be used more than once per media section (including the session-level section). A session update MAY change the direction qualifiers of extensions under use. A session update MAY add or remove extension(s). Identifiers values in the valid range MUST NOT be altered (remapped).

Note that, under this rule, the same local identifier cannot be used for two extensions for the same media, even when one is "sendonly" and the other "recvonly", as it would then be impossible to make either of them sendrecv (since re-numbering is not permitted either).

If a party wishes to offer mutually exclusive alternatives, then multiple extensions with the same identifier in the (unusable) range 4096-4351 MAY be offered; the answerer SHOULD select at most one of the offered extensions with the same identifier, and remap it to a free identifier in the valid range, for that extension to be usable.

Similarly, if more extensions are offered than can be fit in the valid range, identifiers in the range 4096-4351 MAY be offered; the answerer SHOULD choose those that are desired, and remap them to a free identifier in the valid range.

It is always allowed to place the offered identifier value "as is" in the SDP answer (for example, due to lack of a free identifier value in the valid range). Extensions with an identifier outside the valid range MUST NOT, of course, be used. If needed, the offerer or answerer can update the session to make space for such an extension.

Rationale: the range 4096-4351 for these negotiation identifiers is deliberately restricted to allow expansion of the range of valid identifiers in future.

Either party MAY include extensions in the stream other than those negotiated, or those negotiated as "inactive", for example, for the benefit of intermediate nodes. Only extensions that appeared with an identifier in the valid range in SDP originated by the sender can be sent.

Example (port numbers, RTP profiles, payload IDs and rtpmaps, etc. all omitted for brevity):

The offer:

```
a=extmap:1 URI-toffset
a=extmap:14 URI-obscure
a=extmap:4096 URI-gps-string
a=extmap:4096 URI-gps-binary
a=extmap:4097 URI-frametype
m=video
a=sendrecv
m=audio
a=sendrecv
```

The answerer is interested in receiving GPS in string format only on video, but cannot send GPS at all. It is not interested in transmission offsets on audio, and does not understand the URI-obscure extension. It therefore moves the extensions from session level to media level, and adjusts the declarations:

```
m=video
a=sendrecv
a=extmap:1 URI-toffset
a=extmap:2/recvonly URI-gps-string
a=extmap:3 URI-frametype
m=audio
a=sendrecv
a=extmap:1/sendonly URI-toffset
```

8. BNF Syntax

The syntax definition below uses ABNF according to [RFC5234]. The syntax element 'URI' is defined in [RFC3986] (only absolute URIs are permitted here). The syntax element 'extmap' is an attribute as defined in [RFC4566], i.e., "a=" precedes the extmap definition. Specific extensionattributes are defined by the specification that defines a specific extension name; there can be several.

```
extmap = mapentry SP extensionname [SP extensionattributes]

extensionname = URI

direction = "sendonly" / "recvonly" / "sendrecv" / "inactive"

mapentry = "extmap:" 1*5DIGIT ["/" direction]

extensionattributes = byte-string

URI = <Defined in RFC 3986>

byte-string = <Defined in RFC 4566>

SP = <Defined in RFC 5234>

DIGIT = <Defined in RFC 5234>
```

9. Security Considerations

This document defines only a place to transmit information; the security implications of each of the extensions MUST be discussed with those extensions.

Extensions usage is negotiated using [RFC3264] so integrity protection and end-to-end authentication MUST be used. The security considerations of [RFC3264] MUST be followed, to prevent, for example, extension usage blocking.

Header extensions have the same security coverage as the RTP header itself. When Secure Real-time Transport Protocol (SRTP) [RFC3711] is used to protect RTP sessions, the RTP payload can be both encrypted and integrity protected, while the RTP header is either unprotected or integrity protected. In order to prevent DOS attacks, for example, by changing the header extension integrity protection SHOULD be used. Lower layer security protection like DTLS[RFC6347] MAY be used. RTP header extensions can carry sensitive information for which participants in multimedia sessions want confidentiality. RFC6904 [RFC6904] provides a mechanism, extending the mechanisms of SRTP, to selectively encrypt RTP header extensions in SRTP.

Other security options for securing RTP are discussed in [RFC7201].

10. IANA Considerations

This document updates the IANA consideration to reference this document and adds a new SDP attribute in section 10.3

Note to IANA : change RFCxxxx to this RFC number and remove the note.

10.1. Identifier Space for IANA to Manage

The mapping from the naming URI form to a reference to a specification is managed by IANA. Insertion into this registry is under the requirements of "Expert Review" as defined in [RFC5226].

The IANA will also maintain a server that contains all of the registered elements in a publicly accessible space.

Here is the formal declaration to comply with the IETF URN Sub-namespace specification [RFC3553].

- Registry name: RTP Compact Header Extensions
- Specification: RFC 5285 and RFCs updating RFC 5285.
- Information required:

  A. The desired extension naming URI

  B. A formal reference to the publicly available specification

  C. A short phrase describing the function of the extension

  D. Contact information for the organization or person making the registration

  For extensions defined in RFCs, the URI SHOULD be of the form urn:ietf:params:rtp-hdrext:, and the formal reference is the RFC number of the RFC documenting the extension.

- Review process: Expert review is REQUIRED. The expert review SHOULD check the following requirements:

  1. that the specification is publicly available;

  2. that the extension complies with the requirements of RTP and this specification, for extensions;

  3. that the extension specification is technically consistent (in itself and with RTP), complete, and comprehensible;

  4. that the extension does not duplicate functionality in existing IETF specifications (including RTP itself), or other extensions already registered;

5. that the specification contains a security analysis regarding the content of the header extension;

6. that the extension is generally applicable, for example point-to-multipoint safe, and the specification correctly describes limitations if they exist; and

7. that the suggested naming URI form is appropriately chosen and unique.

- Size and format of entries: a mapping from a naming URI string to a formal reference to a publicly available specification, with a descriptive phrase and contact information.

- Initial assignments: none.

10.2. Registration of the SDP extmap Attribute

IANA is requested to register the extmap SDP [RFC4566] attribute.

```
SDP Attribute ("att-field"):
Attribute name:     extmap
Long form:          generic header extension map definition
Type of name:       att-field
Type of attribute:  Media or session level
Subject to charset: No
Purpose:            defines the mapping from the extension numbers used
in
                           packet headers into extension names.
Reference:          [RFCXXXX]
Values:             See [RFCXXXX]
```

10.3. Registration of the SDP extmap-allow-mixed Attribute

The IANA is requested to register one new SDP attribute:

```
SDP Attribute ("att-field"):
Attribute name:     extmap-allow-mixed
Long form:          One and Two bytes mixed mode
Type of name:       att-field
Type of attribute:  Media or session level
Subject to charset: No
Purpose:            Negotiate the use of One and Two bytes
                    in the same RTP stream.
Reference:          [RFCXXXX]
Values:             None
```

11. Changes from RFC5285

The major motivation for updating [RFC5285] was to allow having one byte and two bytes RTP header extensions in the same RTP stream (but not in the same RTP packet). The support for this case is negotiated using a new SDP attribute "extmap-allowed-mixed" specified in this document.

The other major change is to update the requirement from the RTP specification and[RFC5285] that the header extension "is designed so that the header extension MAY be ignored". This is described in section 4.1.

The transmission consideration section (4.1.1) adds more text to clarify when and how many times to send the RTP header extension to provide higher probability of delivery

>The security section was expanded

The rest of the changes are editorial.

12. Acknowledgments

Both Brian Link and John Lazzaro provided helpful comments on an initial draft of this document. Colin Perkins was helpful in reviewing and dealing with the details. The use of URNs for IETF-defined extensions was suggested by Jonathan Lennox, and Pete Cordell was instrumental in improving the padding wording. Dave Oran provided feedback and text in the review. Mike Dolan contributed the two-byte header form. Magnus Westerlund and Tom Taylor were instrumental in managing the registration text.

13. References

13.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

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

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

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

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

13.2. Informative References

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

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

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

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

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

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

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

Authors' Addresses

Roni Even (editor)
Huawei Technologies
Shabazi 12A
Tel Aviv
Israel

Email: Roni.even@mail01.huawei.com

David Singer
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Phone: +1 408 996 1010
Email: singer@apple.com
URI: http://www.apple.com/quicktime

Harikishan Desineni
10001 Pacific Heights Blvd
San Diego, CA 92121
USA

Phone: +1 858 845 8996
Email: hdesinen@quicinc.com

AVTCore
Internet-Draft
Obsoletes: 5285 (if approved)
Intended status: Standards Track
Expires: February 3, 2018

D. Singer
Apple, Inc.
H. Desineni
Qualcomm
R. Even, Ed.
Huawei Technologies

August 2, 2017

A General Mechanism for RTP Header Extensions
draft-ietf-avtcore-rfc5285-bis-14.txt

Abstract

This document provides a general mechanism to use the header extension feature of RTP (the Real-Time Transport Protocol). It provides the option to use a small number of small extensions in each RTP packet, where the universe of possible extensions is large and registration is de-centralized. The actual extensions in use in a session are signaled in the setup information for that session. This document obsoletes RFC5285.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on February 3, 2018.

Table of Contents

1. Introduction

The RTP specification [RFC3550] provides a capability to extend the RTP header. It defines the header extension format and rules for its use in Section 5.3.1. The existing header extension method permits at most one extension per RTP packet, identified by a 16-bit identifier and a 16-bit length field specifying the length of the header extension in 32-bit words.

This mechanism has two conspicuous drawbacks. First, it permits only one header extension in a single RTP packet. Second, the specification gives no guidance as to how the 16-bit header extension identifiers are allocated to avoid collisions.

This specification removes the first drawback by defining a backward-compatible and extensible means to carry multiple header extension elements in a single RTP packet. It removes the second drawback by defining that these extension elements are named by URIs, defining an IANA registry for extension elements defined in IETF specifications, and a Session Description Protocol (SDP) method for mapping between the naming URIs and the identifier values carried in the RTP packets.

This header extension applies to RTP/AVP (the Audio/Visual Profile) and its extensions.

This document obsoletes [RFC5285] and removes a limitation from RFC5285 that did not allow sending both one-byte and two-byte header extensions in the same RTP stream.

2. Requirements Notation

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].

3. Design Goals

The goal of this design is to provide a simple mechanism whereby multiple identified extensions can be used in RTP packets, without the need for formal registration of those extensions but nonetheless avoiding collision.

This mechanism provides an alternative to the practice of burying associated metadata into the media format bit stream. This has often been done in media data sent over fixed-bandwidth channels. Once this is done, a decoder for the specific media format needs to extract the metadata. Also, depending on the media format, the metadata can be added at the time of encoding the media so that the bit-rate used for the metadata is taken into account. But the metadata can be unknown at that time. Inserting metadata at a later time can cause a decode and re-encode to meet bit-rate requirements.

In some cases, a more appropriate, higher-level mechanism may be available, and if so, it can be used. For cases where a higher-level mechanism is not available, it is better to provide a mechanism at the RTP level than have the metadata be tied to a specific form of media data.

4. Packet Design

4.1. General

The following design is fit into the "header extension" of the RTP extension, as described above.

The presence and format of this header extension and its contents are negotiated or defined out-of-band, such as through signaling (see below for SDP signaling). The 16-bit identifier for the two forms of RTP extension defined here is only an architectural constant (e.g., for use by network analyzers); it is the negotiation/definition (e.g., in SDP) that is the definitive indication that this header extension is present.

The RTP specification [RFC3550] states that RTP "is designed so that the header extension may be ignored by other interoperating implementations that have not been extended". The intent of this restriction is that RTP header extensions MUST NOT be used to extend RTP itself in a manner that is backwards incompatible with non-extended implementations. For example, a header extension is not allowed to change the meaning or interpretation of the standard RTP header fields, or of the RTCP Control Protocol (RTCP). Header extensions MAY carry metadata in addition to the usual RTP header information, provided the RTP layer can function if that metadata is missing. For example, RTP header extensions can be used to carry data that’s also sent in RTCP, as an optimisation to lower latency, since they’ll fall back to the original, non-optimised, behaviour if the header extension is not present. The use of header extensions to convey information that will, if missing, disrupt the behaviour of a higher layer application that builds on top of RTP is only acceptable if this doesn’t affect interoperability at the RTP layer. For example, applications that use the SDP BUNDLE extension with the MID RTP header extension [I-D.ietf-mmusic-sdp-bundle-negotiation] to correlate RTP streams with SDP m= lines likely won’t work with full functionality if the MID is missing, but the operation of the RTP layer of those applications will be unaffected. Support for RTP header extensions based on this memo is negotiated using, for example, SDP Offer/Answer [RFC3264]; intermediaries aware of the RTP header extensions are advised to be cautious when removing or generating RTP header extensions see section 4.7 of [RFC7667].

The RTP header extension is formed as a sequence of extension elements, with possible padding. Each extension element has a local identifier and a length. The local identifiers MAY be mapped to a larger namespace in the negotiation (e.g., session signaling).

4.1.1. Transmission Considerations

As is good network practice, data should only be transmitted when needed. The RTP header extension SHOULD only be present in a packet if that packet also contains one or more extension elements, as defined here. An extension element SHOULD only be present in a packet when needed; the signaling setup of extension elements indicates only that those elements can be present in some packets, not that they are in fact present in all (or indeed, any) packets.

Some general considerations for getting the header extensions delivered to the receiver:

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely, what distribution would be needed to avoid getting all repetitions of the header extensions lost in a single burst.

2. If a set of packets are all needed to enable decoding, there is commonly no reason for including the header extension in all of these packets, as they share fate. Instead, at most one instance of the header extension per independently decodable set of media data would be a more efficient use of the bandwidth.

3. How early the Header Extension item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.

4. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

As a summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be optimal. N is selected so that the header extension target delivery probability reaches 1-P^N, where P is the probability of packet loss. For point to point or small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has reached all receivers and stop further repetitions. Feedback that can be used includes the

RTCP XR Loss RLE report block [RFC3611], which will indicate successful delivery of particular packets. If the RTP/AVPF Transport Layer Feedback Messages for generic NACK [RFC4585] is used, it can indicate the failure to deliver an RTP packet with the header extension, thus indicating the need for further repetitions. The normal RTCP report blocks can also provide an indicator of successful delivery, if no losses are indicated for a reporting interval covering the RTP packets with the header extension. Note that loss of an RTCP packet reporting on an interval where RTP header extension packets were sent, does not necessarily mean that the RTP header extension packets themselves were lost.

4.1.2. Header Extension Type Considerations

Each extension element in a packet has a local identifier (ID) and a length. The local identifiers present in the stream MUST have been negotiated or defined out-of-band. There are no static allocations of local identifiers. Each distinct extension MUST have a unique ID. The ID value 0 is reserved for padding and MUST NOT be used as a local identifier.

An extension element with an ID value equal 0 MUST NOT have len field greater than 0. If such an extension element is encountered, its length field MUST be ignored, processing of the entire extension MUST terminate at that point, and only the extension elements present prior to the element with ID 0 and len field greater than 0 SHOULD be considered.

There are two variants of the extension: one-byte and two-byte headers. Since it is expected that (a) the number of extensions in any given RTP session is small and (b) the extensions themselves are small, the one-byte header form is preferred and MUST be supported by all receivers. A stream MUST contain only one-byte or only two-byte headers unless it is known that all recipients support mixing, either by SDP Offer/Answer [RFC3264] negotiation (see section 6) or by out-of-band knowledge. Each RTP packet with an RTP header extension following this specification will indicate if it contains one or two byte header extensions through the use of the "defined by profile" field. Extension element types that do not match the header extension format, i.e. one- or two-byte, MUST NOT be used in that RTP packet. Transmitters SHOULD NOT use the two-byte form when all extensions are small enough for the one-byte header form. Transmitters that intend to send the two-byte form SHOULD negotiate the use of IDs above 14 if they want to let the Receivers know that they intend to use two-byte form, for example if the RTP header extension is longer than 16 bytes. A transmitter may be aware that an intermediary may add RTP header extensions; in this case the transmitter SHOULD use two-byte form.

A sequence of extension elements, possibly with padding, forms the header extension defined in the RTP specification. There are as many extension elements as fit into the length as indicated in the RTP header extension length. Since this length is signaled in full 32-bit words, padding bytes are used to pad to a 32-bit boundary. The entire extension is parsed byte-by-byte to find each extension element (no alignment is needed), and parsing stops at the earlier of the end of the entire header extension, or in "one-byte headers only" case, on encountering an identifier with the reserved value of 15.

In both forms, padding bytes have the value of 0 (zero). They MAY be placed between extension elements, if desired for alignment, or after the last extension element, if needed for padding. A padding byte does not supply the ID of an element, nor the length field. When a padding byte is found, it is ignored and the parser moves on to interpreting the next byte.

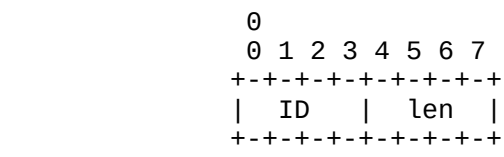
Note carefully that the one-byte header form allows for data lengths between 1 and 16 bytes, by adding 1 to the signaled length value (thus, 0 in the length field indicates 1 byte of data follows). This allows for the important case of 16-byte payloads. This addition is not performed for the two-byte headers, where the length field signals data lengths between 0 and 255 bytes.

Use of RTP header extensions will reduce the efficiency of RTP header compression, since the header extension will be sent uncompressed unless the RTP header compression module is updated to recognize the extension header. If header extensions are present in some packets, but not in others, this can also reduce compression efficiency by requiring an update to the fixed header to be conveyed when header extensions start or stop being sent. The interactions of the RTP header extension and header compression is explored further in [RFC2508] and [RFC3095].

4.2. One-Byte Header

In the one-byte header form of extensions, the 16-bit value required by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", MUST have the fixed bit pattern 0xBEDE (the pattern was picked for the trivial reason that the first version of this specification was written on May 25th the feast day of the Venerable Bede).

Each extension element MUST start with a byte containing an ID and a length:

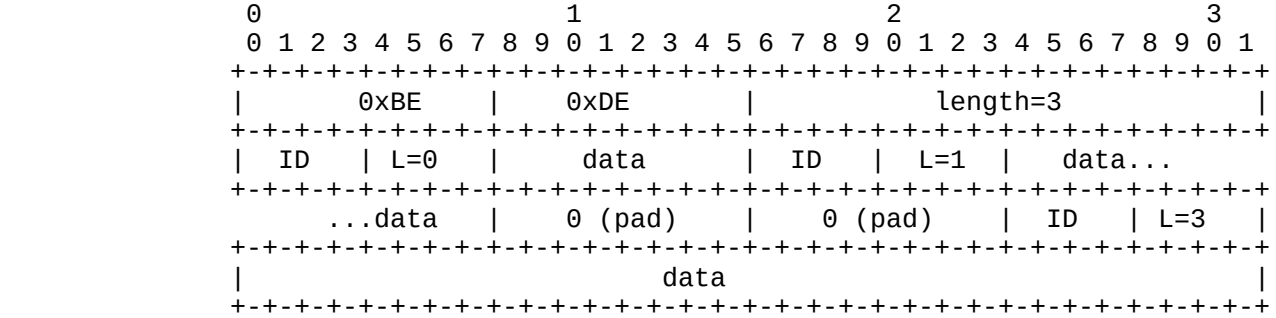


The 4-bit ID is the local identifier of this element in the range 1-14 inclusive. In the signaling section, this is referred to as the valid range.

The local identifier value 15 is reserved for future extension and MUST NOT be used as an identifier. If the ID value 15 is encountered, its length field MUST be ignored, processing of the entire extension MUST terminate at that point, and only the extension elements present prior to the element with ID 15 SHOULD be considered.

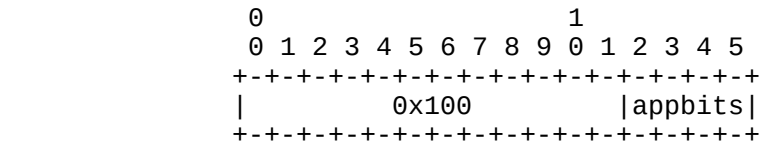
The 4-bit length is the number minus one of data bytes of this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows, and a value of 15 (the maximum) indicates element data of 16 bytes. (This permits carriage of 16-byte values, which is a common length of labels and identifiers, while losing the possibility of zero-length values -- which would often be padded anyway.)

An example header extension, with three extension elements, and some padding follows:



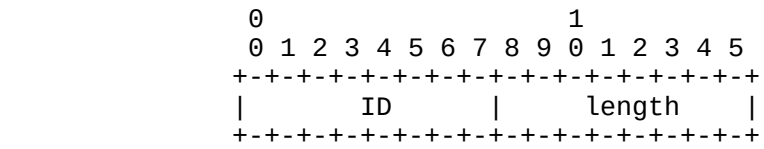
4.3. Two-Byte Header

In the two-byte header form, the 16-bit value defined by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", is defined as shown below.



The appbits field is 4 bits that are application-dependent and MAY be defined to be any value or meaning, and are outside the scope of this specification. For the purposes of signaling, this field is treated as a special extension value assigned to the local identifier 256. If no extension has been specified through configuration or signaling for this local identifier value 256, the appbits field SHOULD be set to all 0s by the sender and MUST be ignored by the receiver.

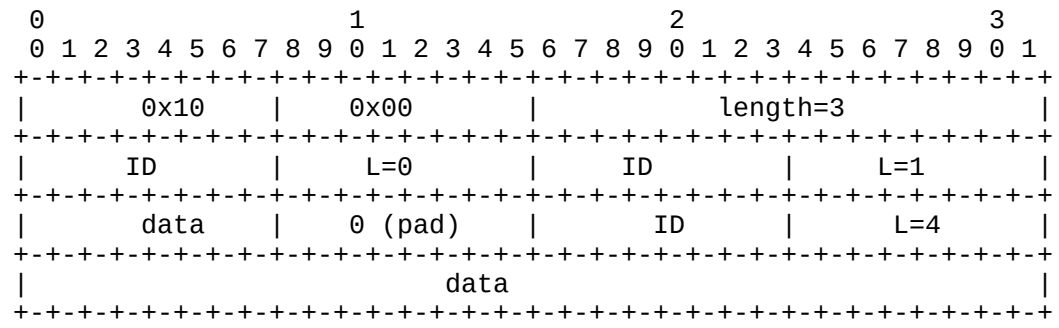
Each extension element starts with a byte containing an ID and a byte containing a length:



The 8-bit ID is the local identifier of this element in the range 1-255 inclusive. In the signaling section, the range 1-256 is referred to as the valid range, with the values 1-255 referring to extension elements, and the value 256 referring to the 4-bit field 'appbits' (above). Note that there is one ID space for both one-byte and two-byte form. This means that the lower values (1-14) can be used in the 4-bit ID field in the one-byte header format with the same meanings.

The 8-bit length field is the length of extension data in bytes not including the ID and length fields. The value zero indicates there is no data following.

An example header extension, with three extension elements, and some padding follows:



5. SDP Signaling Design

The indication of the presence of this extension, and the mapping of local identifiers used in the header extension to a larger namespace, MUST be performed out-of-band, for example, as part of an SDP Offer/Answer [RFC3264]. This section defines such signaling in SDP.

A usable mapping MUST use IDs in the valid range, and each ID in this range MUST be used only once for each media (or only once if the mappings are session level). Mappings that do not conform to these rules MAY be presented, for instance, during SDP Offer/Answer [RFC3264] negotiation as described in the next section, but remapping to conformant values is necessary before they can be applied.

Each extension is named by a URI. That URI MUST be absolute, and precisely identifies the format and meaning of the extension. URIs that contain a domain name SHOULD also contain a month-date in the form mmyyyy. The definition of the element and assignment of the URI MUST have been authorized by the owner of the domain name on or very close to that date. (This avoids problems when domain names change ownership.) If the resource or document defines several extensions, then the URI MUST identify the actual extension in use, e.g., using a fragment or query identifier (characters after a '#' or '?' in the URI).

Rationale: the use of URIs provides for a large, unallocated space, and gives documentation on the extension. The URIs do not have to be de-referencable, in order to permit confidential or experimental use, and to cover the case when extensions continue to be used after the organization that defined them ceases to exist.

An extension URI with the same attributes MUST NOT appear more than once applying to the same stream, i.e., at session level or in the

declarations for a single stream at media level. (The same extension can, of course, be used for several streams, and can appear with different extensionattributes for the same stream.)

For extensions defined in RFCs, the URI used SHOULD be a URN starting "urn:ietf:params:rtp-hdrext:" and followed by a registered, descriptive name.

The registration requirements are detailed in the IANA Considerations section, below.

An example (this is only an example), where 'avt-example-metadata' is the hypothetical name of a header extension, might be:

urn:ietf:params:rtp-hdrext:avt-example-metadata

An example name not from the IETF (this is only an example) might be:

http://example.com/082005/ext.htm#example-metadata

The mapping MAY be provided per media stream (in the media-level section(s) of SDP, i.e., after an "m=" line) or globally for all streams (i.e., before the first "m=" line, at session level). The definitions MUST be either all session level or all media level; it is not permitted to mix the two styles. In addition, as noted above, the IDs used MUST be unique in each media section of the SDP, or unique in the session for session-level SDP declarations.

Each local identifier potentially used in the stream is mapped to an extension identified by a URI using an attribute of the form:

a=extmap:<value>["/"<direction>] <URI> <extensionattributes>

where <URI> is a URI, as above, <value> is the local identifier (ID) of this extension and is an integer in the valid range (0 is reserved for padding in both forms, and 15 is reserved in the one-byte header form, as noted above), and <direction> is one of "sendonly", "recvonly", "sendrecv", or "inactive" (without the quotes) with relation to the device being configured.

The formal BNF syntax is presented in a later section of this specification.

Example:

a=extmap:1 http://example.com/082005/ext.htm#ttime

a=extmap:2/sendrecv http://example.com/082005/ext.htm#xmeta short

When SDP signaling is used for the RTP session, it is the presence of the 'extmap' attribute(s) that is diagnostic that this style of header extensions is used, not the magic number ("BEDE" or "100") indicated above.

6. SDP Signaling for support of mixed one byte and two bytes header extensions.

In order to allow for backward interoperability with systems that do not support mixing of one byte and two bytes header extensions this document defines the "a=extmap-allow-mixed" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the participant is capable of supporting this new mode. The attribute takes no value. This attribute can be used at the session or media levels. A participant that proposes the use of this mode SHALL itself support the reception of mixed one byte and two bytes header extensions.

If SDP Offer/Answer [RFC3264] is supported and used,the negotiation for mixed one byte and two bytes extension MUST be negotiated using SDP Offer/Answer [RFC3264]. In the absence of negotiations using SDP Offer/Answer, for example when declarative SDP is used, mixed headers MUST NOT occur unless the transmitter has some (out of band) knowledge that all potential recipients support this mode.

The formal definition of this attribute is:

Name: extmap-allow-mixed

Value: none

Usage Level: session, media

Charset Dependent: no

Example:

a=extmap-allow-mixed

When doing SDP Offer/Answer [RFC3264] an offering client that wishes to use both one and two bytes extensions MUST include the attribute "a= extmap-allow-mixed " in the SDP offer. If "a= extmap-allow-mixed " is present in the offer SDP, the answerer that supports this mode and wishes to use it SHALL include the "a=extmap-allow-mixed " attribute in the answer. In the cases where the attribute has been excluded, both clients SHALL NOT use mixed one bytes and two bytes extensions in the same RTP stream but MAY use one-byte or two-bytes form exclusively (see section 4.1.2).

When used in [I-D.ietf-mmusic-sdp-bundle-negotiation] this attribute is specified as identical category for the [I-D.ietf-mmusic-sdp-mux-attributes]. This allows for only a subset of the m-lines in the bundle group to offer extmap-allow-mixed. When an answerer supporting the extmap-allow-mix attribute receives an offer where only some of the m-lines in the bundle group include the extmap-allow-mixed attribute, the answerer MUST receive this offer and support mixed one-byte and two-bytes only for those m-lines. Transmitters MUST only send RTP header extensions using mixed on those RTP streams originating from those media sources (m=) blocks that includes extmap-allow-mixed, and are RECOMMENDED to support receiving mixed on all RTP streams being received in an RTP session where at least one bundled m= block is indicating extmap-allow-mixed.

7. SDP Offer/Answer

The simple signaling described above for the extmap attribute MAY be enhanced in an SDP Offer/Answer [RFC3264] context, to permit:

- asymmetric behavior (extensions sent in only one direction),
- the offer of mutually exclusive alternatives, or
- the offer of more extensions than can be sent in a single session.

A direction attribute MAY be included in an extmap; without it, the direction implicitly inherits, of course, from the stream direction, or is "sendrecv" for session-level attributes or extensions of "inactive" streams. The direction MUST be one of "sendonly", "recvonly", "sendrecv", or "inactive" as specified in [RFC3264]

Extensions, with their directions, MAY be signaled for an "inactive" stream. It is an error to use an extension direction incompatible with the stream direction (e.g., a "sendonly" attribute for a "recvonly" stream).

If an offer or answer contains session-level mappings (and hence no media-level mappings), and different behavior is desired for each stream, then the entire set of extension map declarations MAY be moved into the media-level section(s) of the SDP. (Note that this specification does not permit mixing global and local declarations, to make identifier management easier.)

If an extension map is offered as "sendrecv", explicitly or implicitly, and asymmetric behavior is desired, the SDP answer MAY be changed to modify or add direction qualifiers for that extension.

If an extension is marked as "sendonly" and the answerer desires to receive it, the extension MUST be marked as "recvonly" in the SDP answer. An answerer that has no desire to receive the extension or does not understand the extension SHOULD remove it from the SDP answer. An answerer MAY want to respond that he supports the extension and does not want to receive it at the moment but may offer to receive it in a future offer, will mark the extension as "inactive"

If an extension is marked as "recvonly" and the answerer desires to send it, the extension MUST be marked as "sendonly" in the SDP answer. An answerer that has no desire to, or is unable to, send the extension SHOULD remove it from the SDP answer. An answerer MAY want to respond that he support this extension yet has no intention of sending it now but may offer to send it in a future offer by marking the extension as "inactive"

Local identifiers in the valid range inclusive in an offer or answer must not be used more than once per media section (including the session-level section). The local identifiers MUST be unique in an RTP session and the same identifier MUST be used for the same offered extension in the answer. A session update MAY change the direction qualifiers of extensions under use. A session update MAY add or remove extension(s). Identifiers values in the valid range MUST NOT be altered (remapped).

Note that, under this rule, the same local identifier cannot be used for two extensions for the same media, even when one is "sendonly" and the other "recvonly", as it would then be impossible to make either of them sendrecv (since re-numbering is not permitted either).

If a party wishes to offer mutually exclusive alternatives, then multiple extensions with the same identifier in the extended range 4096-4351 MAY be offered; the answerer SHOULD select at most one of the offered extensions with the same identifier, and remap it to a free identifier in the valid range, for that extension to be usable.

Similarly, if more extensions are offered than can be fit in the valid range, identifiers in the range 4096-4351 MAY be offered; the answerer SHOULD choose those that are desired, and remap them to a free identifier in the valid range.

An answerer may copy an extmap for an identifier in the extended range into the answer to indicate to the offerer that it supports that extension. Of course, such an extension cannot be used, since there is no way to specify them in an extension header. If needed, the offerer or answerer can update the session to assign a valid identifier to that extension URI.

Rationale: the range 4096-4351 for these negotiation identifiers is deliberately restricted to allow expansion of the range of valid identifiers in future.

Either party MAY include extensions in the stream other than those negotiated, or those negotiated as "inactive", for example, for the benefit of intermediate nodes. Only extensions that appeared with an identifier in the valid range in SDP originated by the sender can be sent.

Example (port numbers, RTP profiles, payload IDs and rtpmaps, etc. all omitted for brevity):

The offer:

```
a=extmap:1 URI-toffset
a=extmap:14 URI-obscure
a=extmap:4096 URI-gps-string
a=extmap:4096 URI-gps-binary
a=extmap:4097 URI-frametype
m=video
a=sendrecv
m=audio
a=sendrecv
```

The answerer is interested in receiving GPS in string format only on video, but cannot send GPS at all. It is not interested in transmission offsets on audio, and does not understand the URI-obscure extension. It therefore moves the extensions from session level to media level, and adjusts the declarations:

```
m=video
a=sendrecv
a=extmap:1 URI-toffset
a=extmap:2/recvonly URI-gps-string
a=extmap:3 URI-frametype
m=audio
a=sendrecv
a=extmap:1/sendonly URI-toffset
```

When using [I-D.ietf-mmusic-sdp-bundle-negotiation] to bundle multiple m-lines the extmap attribute falls under the special category of [I-D.ietf-mmusic-sdp-mux-attributes]. All the m-lines in a bundle group are considered to be part of the same local identifier (ID) space. If an RTP header extension, i.e. a particular extension URI and configuration using <extensionattributes>, is offered in multiple m-lines that are part of the same bundle group it MUST use the same ID in all of these m-lines. Each m-line in a bundle group

can include different RTP header extensions allowing for example audio and video sources to use different sets of RTP header extensions. It SHALL be assumed that for any RTP header extension, difference in configuration using any of the <extensionattributes> is important and need to be preserved to any receiver, thus requiring assignment of different IDs. Any RTP header extension that does not match this assumption MUST explicitly provide rules for what are compatible configurations that can be sent with the same ID. The directionality of the RTP header extensions in each m-line of the bundle group are handled as the non-bundled case. This allows for specifying different directionality for each of the repeated extension URI in bundled group.

8. BNF Syntax

The syntax definition below uses ABNF according to [RFC5234]. The syntax element 'URI' is defined in [RFC3986] (only absolute URIs are permitted here). The syntax element 'extmap' is an attribute as defined in [RFC4566], i.e., "a=" precedes the extmap definition. Specific extensionattributes are defined by the specification that defines a specific extension name; there can be several.

Name: extmap

Value: extmap-value

Syntax:

```
extmap-value = mapentry SP extensionname
               [SP extensionattributes]

mapentry = "extmap:" 1*5DIGIT ["/" direction]

extensionname = URI

extensionattributes = byte-string

direction = "sendonly" / "recvonly" / "sendrecv" / "inactive"

URI = <Defined in RFC 3986>

byte-string = <Defined in RFC 4566>

SP = <Defined in RFC 5234>

DIGIT = <Defined in RFC 5234>
```

9. Security Considerations

This document defines only a place to transmit information; the security implications of each of the extensions must be discussed with those extensions.

Extensions usage is negotiated using [RFC3264] so integrity protection and end-to-end authentication MUST be implemented. The security considerations of [RFC3264] MUST be followed, to prevent, for example, extension usage blocking.

Header extensions have the same security coverage as the RTP header itself. When Secure Real-time Transport Protocol (SRTP) [RFC3711] is used to protect RTP sessions, the RTP payload can be both encrypted and integrity protected, while the RTP header is either unprotected or integrity protected. In order to prevent DOS attacks, for example, by changing the header extension, integrity protection SHOULD be used. Lower layer security protection like DTLS[RFC6347] MAY be used. RTP header extensions can carry sensitive information for which participants in multimedia sessions want confidentiality. RFC6904 [RFC6904] provides a mechanism, extending the mechanisms of SRTP, to selectively encrypt RTP header extensions in SRTP.

The RTP application designer needs to consider their security needs, that includes cipher strength for SRTP packets in general and what that means for the integrity and confidentiality of the RTP header extensions. As defined by RFC6904 [RFC6904] the encryption stream cipher for the header extension is dependent on the chosen SRTP cipher.

Other security options for securing RTP are discussed in [RFC7201].

10. IANA Considerations

This document updates the IANA consideration to reference this document and adds a new SDP attribute in section 10.3

Note to IANA : change RFCxxxx to this RFC number and remove the note.

10.1. Identifier Space for IANA to Manage

The mapping from the naming URI form to a reference to a specification is managed by IANA. Insertion into this registry is under the requirements of "Expert Review" as defined in [RFC8126].

The IANA will also maintain a server that contains all of the registered elements in a publicly accessible space.

Here is the formal declaration to comply with the IETF URN Sub-namespace specification [RFC3553].

- Registry name: RTP Compact Header Extensions
- Specification: RFC 5285 and RFCs updating RFC 5285.
- Information required:

  A. The desired extension naming URI

  B. A formal reference to the publicly available specification

  C. A short phrase describing the function of the extension

  D. Contact information for the organization or person making the registration

  For extensions defined in RFCs, the URI SHOULD be of the form urn:ietf:params:rtp-hdrext:, and the formal reference is the RFC number of the RFC documenting the extension.

- Review process: Expert review is REQUIRED. The expert review SHOULD check the following requirements:

  1. that the specification is publicly available;

  2. that the extension complies with the requirements of RTP, and this specification, for header extensions (specifically, that the header extension can be ignored or discarded without breaking the RTP layer);

  3. that the extension specification is technically consistent (in itself and with RTP), complete, and comprehensible;

  4. that the extension does not duplicate functionality in existing IETF specifications (including RTP itself), or other extensions already registered;

  5. that the specification contains a security analysis regarding the content of the header extension;

  6. that the extension is generally applicable, for example point-to-multipoint safe, and the specification correctly describes limitations if they exist; and

  7. that the suggested naming URI form is appropriately chosen and unique.

8. That for [I-D.ietf-mmusic-sdp-bundle-negotiation] multiplexed m-lines, any RTP header extension with difference in configurations of <extensionattributes> that do not require assignment of different IDs, MUST explicitly indicate this and provide rules for what are compatible configurations that can be sent with the same ID.

- Size and format of entries: a mapping from a naming URI string to a formal reference to a publicly available specification, with a descriptive phrase and contact information.

- Initial assignments: none.

10.2. Registration of the SDP extmap Attribute

IANA is requested to update the registration of the extmap SDP [RFC4566] attribute.

- Contact Name and email address: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG Attribute Name: extmap

- Attribute Syntax: See section 8 of [RFCXXXX].

- Attribute Semantics: The details of appropriate values are given in [RFC XXXX].

- Usage Level: Media or session level.

- Charset Dependent: No.

- Purpose: defines the mapping from the extension numbers used in packet headers into extension names.

- O/A Procedures: See section 7 of [RFCXXXX].

- Mux Category: Special.

- Reference: [RFCXXXX]

10.3. Registration of the SDP extmap-allow-mixed Attribute

The IANA is requested to register one new SDP attribute:

- Contact Name and email address: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG.

- Attribute Name: extmap-allow-mixed.

- Attribute Syntax: See section 6 of [RFCXXXX].
- Attribute Semantics: See section 6 of [RFCXXXX].
- Attribute Value: None.
- Usage Level: Media or session level.
- Charset Dependent: no.
- Purpose: Negotiate the use of One and Two bytes in the same RTP stream.
- O/A Procedures: See section 6 of [RFCXXXX].
- Mux Category: Identical
- Reference: [RFCXXXX]

11. Changes from RFC5285

The major motivation for updating [RFC5285] was to allow having one byte and two bytes RTP header extensions in the same RTP stream (but not in the same RTP packet). The support for this case is negotiated using a new SDP attribute "extmap-allow-mixed" specified in this document.

The other major change is to update the requirement from the RTP specification [RFC3550] and [RFC5285] that the header extension "is designed so that the header extension MAY be ignored". This is described in section 4.1.

The transmission consideration section (4.1.1) adds more text to clarify when and how many times to send the RTP header extension to provide higher probability of delivery

>The security section was expanded

The rest of the changes are editorial.

12. Acknowledgments

Both Brian Link and John Lazzaro provided helpful comments on an initial draft of this document. Colin Perkins was helpful in reviewing and dealing with the details. The use of URNs for IETF-defined extensions was suggested by Jonathan Lennox, and Pete Cordell was instrumental in improving the padding wording. Dave Oran provided feedback and text in the review. Mike Dolan contributed the

two-byte header form. Magnus Westerlund and Tom Taylor were instrumental in managing the registration text.

13. References

13.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, DOI 10.17487/RFC2508, February 1999, <http://www.rfc-editor.org/info/rfc2508>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <http://www.rfc-editor.org/info/rfc3095>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

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

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

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

13.2. Informative References

[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-38 (work in progress), April 2017.

[I-D.ietf-mmusic-sdp-mux-attributes] Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-16 (work in progress), December 2016.

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

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <http://www.rfc-editor.org/info/rfc3553>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

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

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

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

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

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <http://www.rfc-editor.org/info/rfc7667>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <http://www.rfc-editor.org/info/rfc8126>.

Authors' Addresses

David Singer
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Phone: +1 408 996 1010
Email: singer@apple.com
URI: http://www.apple.com/quicktime

Harikishan Desineni
Qualcomm
10001 Pacific Heights Blvd
San Diego, CA 92121
USA

Phone: +1 858 845 8996
Email: hdesinen@quicinc.com

Roni Even (editor)
Huawei Technologies
Tel Aviv
Israel

Email: Roni.even@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 12, 2017

M. Thomson
E. Rescorla
Mozilla

October 9, 2016

Unknown Key Share Attacks on uses of Transport Layer Security with the Session Description Protocol (SDP)
draft-thomson-avtcore-sdp-uks-00

Abstract

Unknown key-share attacks on the use of Datagram Transport Layer Security for the Secure Real-Time Transport Protocol (DTLS-SRTP) and its use with Web Real-Time Communications (WebRTC) identity assertions are described. Simple mitigation techniques are defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on April 12, 2017.

Table of Contents

1. Introduction

The use of Transport Layer Security (TLS) [RFC5246] with the Session Description Protocol (SDP) [RFC4566] is defined in [RFC4572]. Further use with Datagram Transport Layer Security (DTLS) [RFC6347] and the Secure Real-time Transport Protocol (SRTP) [RFC3711] is defined as DTLS-SRTP [RFC5763].

In these specifications, key agreement is performed using the TLS or DTLS handshaking protocol, with authentication being tied back to the session description (or SDP) through the use of certificate fingerprints. Communication peers check that a hash, or fingerprint, provided in the SDP matches the certificate that is used in the TLS (or DTLS) handshake. This is defined in [RFC4572].

The design of DTLS-SRTP relies on the integrity of the signaling channel. Certificate fingerprints are assumed to be provided by the communicating peers and carried by the signaling channel without being subject to modification. However, this design is vulnerable to an unknown key-share (UKS) attack where a misbehaving endpoint is able to advertise a key that it does not control. This leads to the creation of sessions where peers are confused about the identify of the participants.

An extension to TLS is defined that can be used to mitigate this attack.

A similar attack is possible with sessions that use WebRTC identity (see Section 5.6 of [I-D.ietf-rtcweb-security-arch]). This issue and a mitigation for it is discussed in more detail in Section 4.

2. Unknown Key-Share Attack

In an unknown key-share attack [UKS], a malicious participant in a protocol claims to control a key that is in reality controlled by some other actor. This arises when the identity associated with a key is not properly bound to the key.

In DTLS-SRTP, an endpoint is able to acquire the certificate fingerprint another entity. By advertising that fingerprint in place of one of its own, the malicious endpoint can cause its peer to communicate with a different peer, even though it believes that it is communicating with the malicious endpoint.

When the identity of communicating peers is established by higher-layer signaling constructs, such as those in SIP [RFC4474] or WebRTC [I-D.ietf-rtcweb-security-arch], this allows an attacker to bind their own identity to a session with any other entity.

By substituting the the fingerprint of one peer for its own, an attacker is able to cause a session to be established where one endpoint has an incorrect value for the identity of its peer. However, the peer does not suffer any such confusion, resulting in each peer involved in the session having a different view of the nature of the session.

This attack applies to any communications established based on the "a=fingerprint" SDP attribute [RFC4572].

2.1. Attack Overview

This vulnerability can be used by an attacker to create a call where there is confusion about the communicating endpoints.

A SIP endpoint or WebRTC endpoint that is configured to reuse a certificate can be attacked if it is willing to conduct two concurrent calls, one of which is with an attacker. The attacker can arrange for the victim to incorrectly believe that is calling the attacker when it is in fact calling a second party. The second party correctly believes that it is talking to the victim.

In a related attack, a single call using WebRTC identity can be attacked so that it produces the same outcome. This attack does not require a concurrent call.

2.2. Limits on Attack Feasibility

The use of TLS with SDP depends on the integrity of session signaling. Assuming signaling integrity limits the capabilities of an attacker in several ways. In particular:

1. An attacker can only modify the parts of the session signaling for a session that they are part of, which is limited to their own offers and answers.

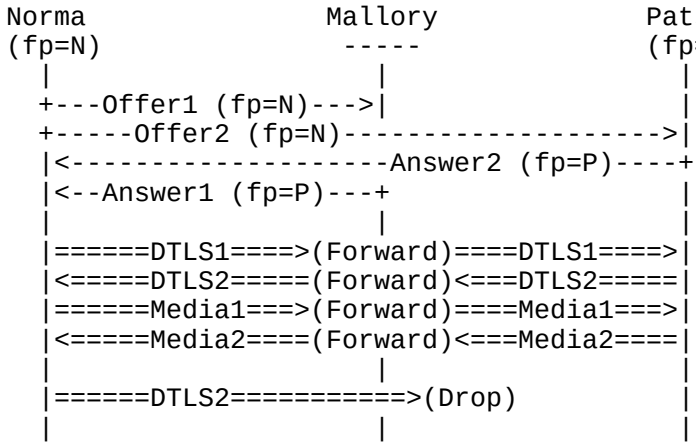
2. No entity will complete communications with a peer unless they are willing to participate in a session with that peer.

The combination of these two constraints make the spectrum of possible attacks quite limited. An attacker is only able to switch its own certificate fingerprint for a valid certificate that is acceptable to its peer. Attacks therefore rely on joining two separate sessions into a single session.

The second condition is not necessary with WebRTC identity if the victim has or is configured with a target peer identity (this is defined in [WEBRTC]). Furthermore, any identity displayed by a browser could be different to the identity used by the application, since the attack affects the browser’s understanding of the peer’s identity.

2.3. Example

In this example, two outgoing sessions are created by the same endpoint. One of those sessions is initiated with the attacker, another session is created toward another honest endpoint. The attacker convinces the endpoint that their session has completed, and that the session with the other endpoint has succeeded.



In this case, Norma is willing to conduct two concurrent sessions. The first session is established with Mallory, who falsely uses Patsy's certificate fingerprint. A second session is initiated between Norma and Patsy. Signaling for both sessions is permitted to complete.

Once complete, the session that is ostensibly between Mallory and Norma is completed by forwarding packets between Norma and Patsy. This requires that Mallory is able to intercept DTLS and media packets from Patsy so that they can be forwarded to Norma at the transport addresses that Norma associates with the first session.

The second session - between Norma and Patsy - is permitted to continue to the point where Patsy believes that it has succeeded. This ensures that Patsy believes that she is communicating with Norma. In the end, Norma believes that she is communicating with Mallory, when she is actually communicating with Patsy.

Though Patsy needs to believe that the second session is successful, Mallory has no real interest in seeing that session complete. Mallory only needs to ensure that Patsy does not abandon the session prematurely. For this reason, it might be necessary to permit the answer from Patsy to reach Norma to allow Patsy to receive a call completion signal, such as a SIP ACK. Once the second session completes, Mallory causes any DTLS packets sent by Norma to Patsy to be dropped.

For the attacked session to be sustained beyond the point that Norma detects errors in the second session, Mallory also needs to block any signaling that Norma might send to Patsy asking for the call to be abandoned. Otherwise, Patsy might receive a notice that the call is failed and thereby abort the call.

This attack creates an asymmetry in the beliefs about the identity of peers. However, this attack is only possible if the victim (Norma) is willing to conduct two sessions concurrently, and if the same certificate - and therefore "a=fingerprint" value - is used in both sessions.

2.4. Interactions with Key Continuity

Systems that use key continuity might be able to detect an unknown key-share attack if a session with the actual peer (i.e., Patsy in the example) was established in the past. Whether this is possible depends on how key continuity is implemented.

Implementations that maintain a single database of identities with an index on peer keys could discover that the identity saved for the peer key does not match the claimed identity. Such an implementation could notice the disparity between the actual keys (Patsy) and the expected keys (Mallory).

In comparison, implementations that first match based on peer identity could treat an unknown key-share attack as though their peer had used a newly-configured device. The apparent addition of a new device could generate user-visible notices (e.g., "Mallory appears to have a new device"). However, such an event is not always considered alarming; some implementations might silently save a new key.

3. Adding a Session Identifier

An attack on DTLS-SRTP is possible because the identity of peers involved is not established prior to establishing the call. Endpoints use certificate fingerprints as a proxy for authentication, but as long as fingerprints are used in multiple calls, they are vulnerable to attacks of the sort described.

The solution to this problem is to assign a new identifier to communicating peers. Each endpoint assigns their peer a unique identifier during call signaling. The peer echoes that identifier in the TLS handshake, binding that identity into the session. Including this new identity in the TLS handshake means that it will be covered by the TLS Finished message, which is necessary to authenticate it (see [SIGMA]). Validating that peers use the correct identifier then means that the session is established between the correct two endpoints.

Rather than define a new identifier and means for signaling it, the "sess-id" field of the o= line in the SDP is used. This field is already required to be unique, thus, no two offers or answers from the same client will have the same value.

The "sess-id" is defined as a decimal sequence [RFC4566]. [RFC3264] subsequently limits "sess-id" to a 63-bit value. Endpoints MUST include a unique 63-bit value in every session description (offer or answer) they generate. Endpoints SHOULD generate this value using a cryptographically-secure random process [RFC4086].

Note: We could define a new attribute for this purpose, but that just makes things harder to deploy. This design limits the protocol changes to the TLS extension and its validation.

A new "sdp_session_id" extension is added to the TLS or DTLS handshake for connections that are established as part of the same call or real-time session.

3.1. The sdp_session_id TLS Extension

The "sdp_session_id" TLS extension carries the unique identifier that an endpoint selects. The value includes the "sess-id" field from the SDP that the endpoint generated when negotiating the session.

The "extension_data" for the "sdp_session_id" extension contains a SdpSessionId struct, described below using the syntax defined in [RFC5246]:

```
struct {
   uint64 sess_id;
   uint16 m_line;
} SdpSessionId;
```

The "sess_id" field of the extension includes the value of the "sess-id" field from the "o=" line of the SDP offer or answer that the endpoint generates.

The "m_line" field of the extension includes the index of the "m=" section of the session description that the TLS connection is generated for, starting at index 0. Bundled media sections [I-D.ietf-mmusic-sdp-bundle-negotiation] are identified by the index of the "m=" section associated with the Answerer BUNDLE-tag. This prevents an attacker from rearranging "m=" sections within the same session.

Where RTP and RTCP [RFC3550] are not multiplexed, it is possible that the two separate DTLS connections carrying RTP and RTCP can be switched. This is considered benign since these protocols are often distinguishable. RTP/RTCP multiplexing is advised to address this problem.

The "sdp_session_id" extension is included in a ClientHello and either ServerHello (for TLS and DTLS versions less than 1.3) or EncryptedExtensions (for TLS 1.3). In TLS 1.3, the extension MUST NOT be included in a ServerHello.

Endpoints MUST check that the "sess_id" parameter in the extension that they receive includes the "sess-id" value that they received in their peer's session description. Endpoints MUST also check that the "m_line" parameter matches their expectations. An endpoint that has receives a "sdp_session_id" extension that is not identical to the value that it expects MUST abort the connection with a fatal "handshake_failure" alert.

An endpoint that is communicating with a peer that does not support this extension will receive a ClientHello, ServerHello or EncryptedExtensions that does not include this extension. An endpoint MAY choose to continue a session without this extension in order to interoperate with peers that do not implement this specification.

In TLS 1.3, the "sdp_session_id" extension MUST be sent in the EncryptedExtensions message.

4. WebRTC Identity Binding

The identity assertion used for WebRTC [I-D.ietf-rtcweb-security-arch] is bound only to the certificate fingerprint of an endpoint and can therefore be copied by an attacker along with the "a=fingerprint" attributes.

The problem is compounded by the fact that an identity provider is not required to verify that the entity requesting an identity assertion controls the keys. Nor is it currently able to perform this validation. Note however that this verification is not a necessary condition for a secure protocol, as established in [SIGMA].

A simple solution to this problem is suggested by [SIGMA]. The identity of endpoints is included under a message authentication code (MAC) during the cryptographic handshake. Endpoints are then expected to validate that their peer has provided an identity that matches their expectations.

In TLS, the Finished message provides a MAC over the entire handshake, so that including the identity in a TLS extension is sufficient to implement this solution. Rather than include a complete identity assertion, a hash of the identity assertion is included in a TLS extension. Peers then need only validate that the

extension contains a hash of the identity assertion they received in signaling in addition to validating the identity assertion.

Endpoints MAY use the "sdp_session_id" extension in addition to this so that two calls between the same parties can't be altered by an attacker.

4.1. The webrtc_id_hash TLS Extension

The "webrtc_id_hash" TLS extension carries a hash of the identity assertion that communicating peers have exchanged.

The "extension_data" for the "webrtc_id_hash" extension contains a WebrtcIdentityHash struct, described below using the syntax defined in [RFC5246]:

```
struct {
   opaque assertion_hash[32];
} WebrtcIdentityHash;
```

A WebRTC identity assertion is provided as a JSON [RFC7159] object that is encoded into a JSON text. The resulting string is then encoded using UTF-8 [RFC3629]. The content of the "webrtc_id_hash" extension are produced by hashing the resulting octets with SHA-256 [FIPS180-2]. This produces the 32 octets of the assertion_hash parameter, which is the sole contents of the extension.

The "a=identity" attribute includes the base64 [RFC4648] encoding of the same octets that were input to the hash. The "webrtc_id_hash" extension is validated by performing base64 decoding on the value of the "a=identity" attribute, hashing the resulting octets using SHA-256, and comparing the results with the content of the extension.

Identity assertions might be provided by only one peer. An endpoint that does not produce an identity assertion MUST generate an empty "webrtc_id_hash" extension in its ClientHello. This allows its peer to include a hash of its identity assertion. An endpoint without an identity assertion MUST omit the "webrtc_id_hash" extension from its ServerHello or EncryptedExtensions message.

A peer that receives a "webrtc_id_hash" extension that is not equal to the value of the identity assertion from its peer MUST immediately fail the TLS handshake with an error. This includes cases where the "a=identity" attribute is not present in the SDP.

A peer that receives an identity assertion, but does not receive a "webrtc_id_hash" extension MAY choose to fail the connection, though it is expected that implementations that were written prior to the

existence of this document will not support these extensions for some time.

In TLS 1.3, the "webrtc_id_hash" extension MUST be sent in the EncryptedExtensions message.

5. Session Concatenation

Use of session identifiers does not prevent an attacker from establishing two concurrent sessions with different peers and forwarding signaling from those peers to each other. Concatenating two signaling sessions creates a situation where both peers believe that they are talking to the attacker when they are talking to each other.

Session concatention is possible at higher layers: an attacker can establish two independent sessions and simply forward any data it receives from one into the other. This kind of attack is prevented by systems that enable peer authentication such as WebRTC identity [I-D.ietf-rtcweb-security-arch] or SIP identity [RFC4474].

In the absence of any higher-level concept of peer identity, the use of session identifiers does not prevent session concatenation. The value to an attacker is limited unless information from the TLS connection is extracted and used with the signaling. For instance, a key exporter [RFC5705] might be used to create a shared secret or unique identifier that is used in a secondary protocol.

If a secondary protocol uses the signaling channel with the assumption that the signaling and TLS peers are the same then that protocol is vulnerable to attack. The identity of the peer at the TLS layer is not guranteed to be the same as the identity of the signaling peer.

It is important to note that multiple connections can be created within the same signaling session. An attacker can concatenate only part of a session, choosing to terminate some connections (and optionally forward data) while arranging to have peers interact directly for other connections. It is even possible to have different peers interact for each connection. This means that the actual identity of the peer for one connection might differ from the peer on another connection.

Information extracted from a TLS connection therefore MUST NOT be used in a secondary protocol outside of that connection if that protocol relies on the signaling protocol having the same peers. Similarly, data from one TLS connection MUST NOT be used in other TLS

connections even if they are established as a result of the same signaling session.

6. Security Considerations

This entire document contains security considerations.

7. IANA Considerations

This document registers two extensions in the TLS "ExtensionType Values" registry established in [RFC5246]:

- The "sdp_session_id" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

- The "webrtc_id_hash" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

8. References

8.1. Normative References

[FIPS180-2] Department of Commerce, National., "NIST FIPS 180-2, Secure Hash Standard", August 2002.

[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-33 (work in progress), October 2016.

[I-D.ietf-rtcweb-security-arch] Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-12 (work in progress), June 2016.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

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

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, DOI 10.17487/RFC4572, July 2006, <http://www.rfc-editor.org/info/rfc4572>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

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

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

8.2. Informative References

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[SIGMA] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols", Annual International Cryptology Conference, Springer, pp. 400-425 , 2003.

[UKS] Blake-Wilson, S. and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", Lecture Notes in Computer Science 1560, Springer, pp. 154-170 , 1999.

[WEBRTC] Bergkvist, A., Burnett, D., Narayanan, A., Jennings, C., and B. Aboba, "WebRTC 1.0: Real-time Communication Between Browsers", W3C WD-webrtc-30160531 , May 2016.

Appendix A. Acknowledgements

This problem would not have been discovered if it weren't for discussions with Sam Scott, Hugo Krawczyk, and Richard Barnes. A solution similar to the one presented here was first proposed by Karthik Bhargavan who provided valuable input on this document. Thyla van der Merwe assisted with a formal model of the solution. Adam Roach provided useful input.

Authors' Addresses

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

Eric Rescorla
Mozilla

Email: ekr@rftm.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

M. Thomson
E. Rescorla
Mozilla

March 13, 2017

Unknown Key Share Attacks on uses of Transport Layer Security with the Session Description Protocol (SDP)
draft-thomson-avtcore-sdp-uks-01

Abstract

Unknown key-share attacks on the use of Datagram Transport Layer Security for the Secure Real-Time Transport Protocol (DTLS-SRTP) and its use with Web Real-Time Communications (WebRTC) identity assertions are described. Simple mitigation techniques are defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Introduction

The use of Transport Layer Security (TLS) [RFC5246] with the Session Description Protocol (SDP) [RFC4566] is defined in [RFC4572]. Further use with Datagram Transport Layer Security (DTLS) [RFC6347] and the Secure Real-time Transport Protocol (SRTP) [RFC3711] is defined as DTLS-SRTP [RFC5763].

In these specifications, key agreement is performed using the TLS or DTLS handshaking protocol, with authentication being tied back to the session description (or SDP) through the use of certificate fingerprints. Communication peers check that a hash, or fingerprint, provided in the SDP matches the certificate that is used in the TLS (or DTLS) handshake. This is defined in [RFC4572].

The design of DTLS-SRTP relies on the integrity of the signaling channel. Certificate fingerprints are assumed to be provided by the communicating peers and carried by the signaling channel without being subject to modification. However, this design is vulnerable to an unknown key-share (UKS) attack where a misbehaving endpoint is able to advertise a key that it does not control. This leads to the creation of sessions where peers are confused about the identify of the participants.

An extension to TLS is defined that can be used to mitigate this attack.

A similar attack is possible with sessions that use WebRTC identity (see Section 5.6 of [I-D.ietf-rtcweb-security-arch]). This issue and a mitigation for it is discussed in more detail in Section 4.

2. Unknown Key-Share Attack

In an unknown key-share attack [UKS], a malicious participant in a protocol claims to control a key that is in reality controlled by some other actor. This arises when the identity associated with a key is not properly bound to the key.

In DTLS-SRTP, an endpoint is able to acquire the certificate fingerprint another entity. By advertising that fingerprint in place of one of its own, the malicious endpoint can cause its peer to communicate with a different peer, even though it believes that it is communicating with the malicious endpoint.

When the identity of communicating peers is established by higher-layer signaling constructs, such as those in SIP [RFC4474] or WebRTC [I-D.ietf-rtcweb-security-arch], this allows an attacker to bind their own identity to a session with any other entity.

By substituting the the fingerprint of one peer for its own, an attacker is able to cause a session to be established where one endpoint has an incorrect value for the identity of its peer. However, the peer does not suffer any such confusion, resulting in each peer involved in the session having a different view of the nature of the session.

This attack applies to any communications established based on the SDP "fingerprint" attribute [RFC4572].

2.1. Attack Overview

This vulnerability can be used by an attacker to create a call where there is confusion about the communicating endpoints.

A SIP endpoint or WebRTC endpoint that is configured to reuse a certificate can be attacked if it is willing to conduct two concurrent calls, one of which is with an attacker. The attacker can arrange for the victim to incorrectly believe that is calling the attacker when it is in fact calling a second party. The second party correctly believes that it is talking to the victim.

In a related attack, a single call using WebRTC identity can be attacked so that it produces the same outcome. This attack does not require a concurrent call.

2.2. Limits on Attack Feasibility

The use of TLS with SDP depends on the integrity of session signaling. Assuming signaling integrity limits the capabilities of an attacker in several ways. In particular:

1. An attacker can only modify the parts of the session signaling for a session that they are part of, which is limited to their own offers and answers.

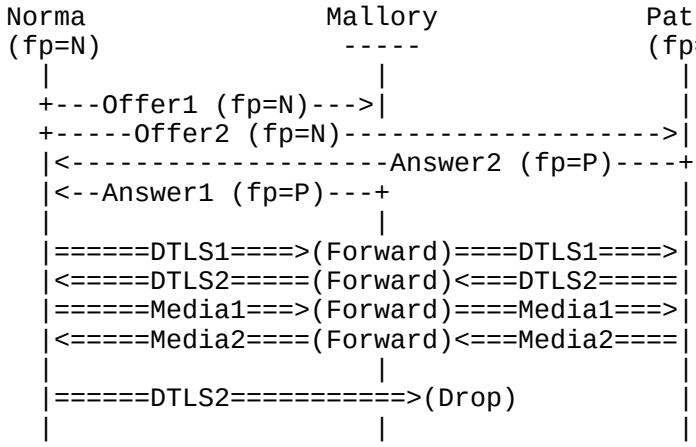
2. No entity will complete communications with a peer unless they are willing to participate in a session with that peer.

The combination of these two constraints make the spectrum of possible attacks quite limited. An attacker is only able to switch its own certificate fingerprint for a valid certificate that is acceptable to its peer. Attacks therefore rely on joining two separate sessions into a single session.

The second condition is not necessary with WebRTC identity if the victim has or is configured with a target peer identity (this is defined in [WEBRTC]). Furthermore, any identity displayed by a browser could be different to the identity used by the application, since the attack affects the browser’s understanding of the peer’s identity.

2.3. Example

In this example, two outgoing sessions are created by the same endpoint. One of those sessions is initiated with the attacker, another session is created toward another honest endpoint. The attacker convinces the endpoint that their session has completed, and that the session with the other endpoint has succeeded.



In this case, Norma is willing to conduct two concurrent sessions. The first session is established with Mallory, who falsely uses Patsy's certificate fingerprint. A second session is initiated between Norma and Patsy. Signaling for both sessions is permitted to complete.

Once complete, the session that is ostensibly between Mallory and Norma is completed by forwarding packets between Norma and Patsy. This requires that Mallory is able to intercept DTLS and media packets from Patsy so that they can be forwarded to Norma at the transport addresses that Norma associates with the first session.

The second session - between Norma and Patsy - is permitted to continue to the point where Patsy believes that it has succeeded. This ensures that Patsy believes that she is communicating with Norma. In the end, Norma believes that she is communicating with Mallory, when she is actually communicating with Patsy.

Though Patsy needs to believe that the second session is successful, Mallory has no real interest in seeing that session complete. Mallory only needs to ensure that Patsy does not abandon the session prematurely. For this reason, it might be necessary to permit the answer from Patsy to reach Norma to allow Patsy to receive a call completion signal, such as a SIP ACK. Once the second session completes, Mallory causes any DTLS packets sent by Norma to Patsy to be dropped.

For the attacked session to be sustained beyond the point that Norma detects errors in the second session, Mallory also needs to block any signaling that Norma might send to Patsy asking for the call to be abandoned. Otherwise, Patsy might receive a notice that the call is failed and thereby abort the call.

This attack creates an asymmetry in the beliefs about the identity of peers. However, this attack is only possible if the victim (Norma) is willing to conduct two sessions concurrently, and if the same certificate - and therefore SDP "fingerprint" attribute value - is used in both sessions.

2.4. Interactions with Key Continuity

Systems that use key continuity might be able to detect an unknown key-share attack if a session with the actual peer (i.e., Patsy in the example) was established in the past. Whether this is possible depends on how key continuity is implemented.

Implementations that maintain a single database of identities with an index on peer keys could discover that the identity saved for the peer key does not match the claimed identity. Such an implementation could notice the disparity between the actual keys (Patsy) and the expected keys (Mallory).

In comparison, implementations that first match based on peer identity could treat an unknown key-share attack as though their peer had used a newly-configured device. The apparent addition of a new device could generate user-visible notices (e.g., "Mallory appears to have a new device"). However, such an event is not always considered alarming; some implementations might silently save a new key.

3. Adding a Session Identifier

An attack on DTLS-SRTP is possible because the identity of peers involved is not established prior to establishing the call. Endpoints use certificate fingerprints as a proxy for authentication, but as long as fingerprints are used in multiple calls, they are vulnerable to attacks of the sort described.

The solution to this problem is to assign a new identifier to communicating peers. Each endpoint assigns their peer a unique identifier during call signaling. The peer echoes that identifier in the TLS handshake, binding that identity into the session. Including this new identity in the TLS handshake means that it will be covered by the TLS Finished message, which is necessary to authenticate it (see [SIGMA]). Validating that peers use the correct identifier then means that the session is established between the correct two endpoints.

This solution relies on the unique identifier given to DTLS sessions using the SDP "dtls-id" attribute [I-D.ietf-mmusic-dtls-sdp]. This field is already required to be unique. Thus, no two offers or answers from the same client will have the same value.

A new "sdp_dtls_id" extension is added to the TLS or DTLS handshake for connections that are established as part of the same call or real-time session. This carries the value of the "dtls-id" attribute and provides integrity protection for its exchange as part of the TLS or DTLS handshake.

3.1. The sdp_dtls_id TLS Extension

The "sdp_dtls_id" TLS extension carries the unique identifier that an endpoint selects. The value includes the "sess-id" field from the SDP that the endpoint generated when negotiating the session.

The "extension_data" for the "sdp_dtls_id" extension contains a SdpDtlsId struct, described below using the syntax defined in [RFC5246]:

```
struct {
   opaque dtls_id<1..255>;
} SdpDtlsId;
```

The "dtls_id" field of the extension includes the value of the "dtls-id" SDP attribute as defined in [I-D.ietf-mmusic-dtls-sdp] (that is, the "dtls-id-value" ABNF production). The value of the "dtls-id" attribute is encoded using ASCII [RFC0020].

Where RTP and RTCP [RFC3550] are not multiplexed, it is possible that the two separate DTLS connections carrying RTP and RTCP can be switched. This is considered benign since these protocols are often distinguishable. RTP/RTCP multiplexing is advised to address this problem.

The "sdp_dtls_id" extension is included in a ClientHello and either ServerHello (for TLS and DTLS versions less than 1.3) or EncryptedExtensions (for TLS 1.3). In TLS 1.3, the extension MUST NOT be included in a ServerHello.

Endpoints MUST check that the "dtls_id" parameter in the extension that they receive includes the "dtls-id" attribute value that they received in their peer’s session description. Comparison can be performed with either the decoded ASCII string or the encoded octets. An endpoint that receives a "sdp_dtls_id" extension that is not identical to the value that it expects MUST abort the connection with a fatal "handshake_failure" alert.

An endpoint that is communicating with a peer that does not support this extension will receive a ClientHello, ServerHello or EncryptedExtensions that does not include this extension. An endpoint MAY choose to continue a session without this extension in

order to interoperate with peers that do not implement this specification.

In TLS 1.3, the "sdp_dtls_id" extension MUST be sent in the EncryptedExtensions message.

4. WebRTC Identity Binding

The identity assertion used for WebRTC [I-D.ietf-rtcweb-security-arch] is bound only to the certificate fingerprint of an endpoint and can therefore be copied by an attacker along with any SDP "fingerprint" attributes.

The problem is compounded by the fact that an identity provider is not required to verify that the entity requesting an identity assertion controls the keys. Nor is it currently able to perform this validation. Note however that this verification is not a necessary condition for a secure protocol, as established in [SIGMA].

A simple solution to this problem is suggested by [SIGMA]. The identity of endpoints is included under a message authentication code (MAC) during the cryptographic handshake. Endpoints are then expected to validate that their peer has provided an identity that matches their expectations.

In TLS, the Finished message provides a MAC over the entire handshake, so that including the identity in a TLS extension is sufficient to implement this solution. Rather than include a complete identity assertion, a collision-resistant hash of the identity assertion is included in a TLS extension. Peers then need only validate that the extension contains a hash of the identity assertion they received in signaling in addition to validating the identity assertion.

Endpoints MAY use the "sdp_dtls_id" extension in addition to this so that two calls between the same parties can’t be altered by an attacker.

4.1. The webrtc_id_hash TLS Extension

The "webrtc_id_hash" TLS extension carries a hash of the identity assertion that communicating peers have exchanged.

The "extension_data" for the "webrtc_id_hash" extension contains a WebrtcIdentityHash struct, described below using the syntax defined in [RFC5246]:

```
struct {
   opaque assertion_hash[32];
} WebrtcIdentityHash;
```

A WebRTC identity assertion is provided as a JSON [RFC7159] object that is encoded into a JSON text. The resulting string is then encoded using UTF-8 [RFC3629]. The content of the "webrtc_id_hash" extension are produced by hashing the resulting octets with SHA-256 [FIPS180-2]. This produces the 32 octets of the assertion_hash parameter, which is the sole contents of the extension.

The SDP "identity" attribute includes the base64 [RFC4648] encoding of the same octets that were input to the hash. The "webrtc_id_hash" extension is validated by performing base64 decoding on the value of the SDP "identity" attribute, hashing the resulting octets using SHA-256, and comparing the results with the content of the extension.

Identity assertions might be provided by only one peer. An endpoint that does not produce an identity assertion MUST generate an empty "webrtc_id_hash" extension in its ClientHello. This allows its peer to include a hash of its identity assertion. An endpoint without an identity assertion MUST omit the "webrtc_id_hash" extension from its ServerHello or EncryptedExtensions message.

A peer that receives a "webrtc_id_hash" extension that is not equal to the value of the identity assertion from its peer MUST immediately fail the TLS handshake with an error. This includes cases where the "a=identity" attribute is not present in the SDP.

A peer that receives an identity assertion, but does not receive a "webrtc_id_hash" extension MAY choose to fail the connection, though it is expected that implementations that were written prior to the existence of this document will not support these extensions for some time.

In TLS 1.3, the "webrtc_id_hash" extension MUST be sent in the EncryptedExtensions message.

5. Session Concatenation

Use of session identifiers does not prevent an attacker from establishing two concurrent sessions with different peers and forwarding signaling from those peers to each other. Concatenating two signaling sessions creates a situation where both peers believe that they are talking to the attacker when they are talking to each other.

Session concatention is possible at higher layers: an attacker can establish two independent sessions and simply forward any data it receives from one into the other. This kind of attack is prevented by systems that enable peer authentication such as WebRTC identity [I-D.ietf-rtcweb-security-arch] or SIP identity [RFC4474].

In the absence of any higher-level concept of peer identity, the use of session identifiers does not prevent session concatenation. The value to an attacker is limited unless information from the TLS connection is extracted and used with the signaling. For instance, a key exporter [RFC5705] might be used to create a shared secret or unique identifier that is used in a secondary protocol.

If a secondary protocol uses the signaling channel with the assumption that the signaling and TLS peers are the same then that protocol is vulnerable to attack. The identity of the peer at the TLS layer is not guranteed to be the same as the identity of the signaling peer.

It is important to note that multiple connections can be created within the same signaling session. An attacker can concatenate only part of a session, choosing to terminate some connections (and optionally forward data) while arranging to have peers interact directly for other connections. It is even possible to have different peers interact for each connection. This means that the actual identity of the peer for one connection might differ from the peer on another connection.

Information extracted from a TLS connection therefore MUST NOT be used in a secondary protocol outside of that connection if that protocol relies on the signaling protocol having the same peers. Similarly, data from one TLS connection MUST NOT be used in other TLS connections even if they are established as a result of the same signaling session.

6. Security Considerations

This entire document contains security considerations.

7. IANA Considerations

This document registers two extensions in the TLS "ExtensionType Values" registry established in [RFC5246]:

- The "sdp_dtls_id" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

- The "webrtc_id_hash" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

8. References

8.1. Normative References

[FIPS180-2] Department of Commerce, National., "NIST FIPS 180-2, Secure Hash Standard", August 2002.

[I-D.ietf-mmusic-dtls-sdp] Holmberg, C. and R. Shpount, "Using the SDP Offer/Answer Mechanism for DTLS", draft-ietf-mmusic-dtls-sdp-21 (work in progress), March 2017.

[I-D.ietf-rtcweb-security-arch] Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-12 (work in progress), June 2016.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

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

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

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, DOI 10.17487/RFC4572, July 2006, <http://www.rfc-editor.org/info/rfc4572>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

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

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

8.2. Informative References

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[SIGMA] Krawczyk, H., "SIGMA: The ’SIGn-and-MAc’approach to authenticated Diffie-Hellman and its use in the IKE protocols", Annual International Cryptology Conference, Springer, pp. 400-425 , 2003.

[UKS] Blake-Wilson, S. and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", Lecture Notes in Computer Science 1560, Springer, pp. 154-170 , 1999.

[WEBRTC] Bergkvist, A., Burnett, D., Narayanan, A., Jennings, C., and B. Aboba, "WebRTC 1.0: Real-time Communication Between Browsers", W3C WD-webrtc-30160531 , May 2016.

Appendix A. Acknowledgements

This problem would not have been discovered if it weren't for discussions with Sam Scott, Hugo Krawczyk, and Richard Barnes. A solution similar to the one presented here was first proposed by Karthik Bhargavan who provided valuable input on this document. Thyla van der Merwe assisted with a formal model of the solution. Adam Roach and Paul E. Jones provided useful input.

Authors' Addresses

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

Eric Rescorla
Mozilla

Email: ekr@rftm.com