AVTCore
Internet-Draft
Intended status: Standards Track
Expires: May 17, 2017

W. Kim
J. Lee
J. Park
D. Kwon
NSRI
D. Kim
Kookmin Univ.

November 13, 2016

The Addition of SRTP crypto suites based on the ARIA algorithms to the SDP Security Descriptions
draft-ietf-avtcore-aria-sdes-02

Abstract

This document defines SRTP crypto suites based on the ARIA block cipher algorithm for use with the Session Description Protocol (SDP) security descriptions.

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 17, 2017.

Table of Contents

1. Introduction

This document defines the SDP Security Descriptions attributes [RFC4568] corresponding to the ARIA transforms which are defined in [I-D.ietf-avtcore-aria-srtp].

1.1. ARIA

ARIA is a general-purpose block cipher algorithm developed by Korean cryptographers in 2003. It is an iterated block cipher with 128-, 192-, and 256-bit keys and encrypts 128-bit blocks in 12, 14, and 16 rounds, depending on the key size. It is secure and suitable for most software and hardware implementations on 32-bit and 8-bit processors. It was established as a Korean standard block cipher algorithm in 2004 [ARIAKS] and has been widely used in Korea, especially for government-to-public services. It was included in PKCS #11 in 2007 [ARIAPKCS]. The algorithm specification and object identifiers are described in [RFC5794].

1.2. SRTP Crypto Suites

The transforms based on ARIA and the corresponding SRTP protection profiles for DTLS-SRTP are defined in [I-D.ietf-avtcore-aria-srtp]. The SDP Security Descriptions [RFC4568] crypto suites corresponding to ARIA transforms [I-D.ietf-avtcore-aria-srtp] are sets as shown in Table 1.

Name	Enc. Key Length	Auth. Tag Length
ARIA_128_CTR_HMAC_SHA1_80	16 octets	10 octets
ARIA_128_CTR_HMAC_SHA1_32	16 octets	4 octets
ARIA_192_CTR_HMAC_SHA1_80	24 octets	10 octets
ARIA_192_CTR_HMAC_SHA1_32	24 octets	4 octets
ARIA_256_CTR_HMAC_SHA1_80	32 octets	10 octets
ARIA_256_CTR_HMAC_SHA1_32	32 octets	4 octets
AEAD_ARIA_128_GCM	16 octets	16 octets
AEAD_ARIA_256_GCM	32 octets	16 octets

Table 1: ARIA Crypto Suites for SRTP

1.3. Terminology

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

2. Parameters

The parameters in each crypto suite listed in Table 1 are described for use with the SDP Security Descriptions attributes [RFC4568].

Parameter	Value
Master key length	128 bits
Master salt length	112 bits
Key Derivation Function	ARIA_128_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_128_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	80 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 2: The ARIA_128_CTR_HMAC_SHA1_80 Crypto Suite

Parameter	Value
Master key length	128 bits
Master salt length	112 bits
Key Derivation Function	ARIA_128_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_128_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	32 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 3: The ARIA_128_CTR_HMAC_SHA1_32 Crypto Suite

Parameter	Value
Master key length	192 bits
Master salt length	112 bits
Key Derivation Function	ARIA_192_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_192_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	80 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 4: The ARIA_192_CTR_HMAC_SHA1_80 Crypto Suite

Parameter	Value
Master key length	192 bits
Master salt length	112 bits
Key Derivation Function	ARIA_192_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_192_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	32 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 5: The ARIA_192_CTR_HMAC_SHA1_32 Crypto Suite

Parameter	Value
Master key length	256 bits
Master salt length	112 bits
Key Derivation Function	ARIA_256_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_256_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	80 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 6: The ARIA_256_CTR_HMAC_SHA1_80 Crypto Suite

Parameter	Value
Master key length	256 bits
Master salt length	112 bits
Key Derivation Function	ARIA_256_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	ARIA_256_CTR
SRTP authentication function	HMAC-SHA1
SRTP authentication key length	160 bits
SRTP authentication tag length	32 bits
SRTCP authentication function	HMAC-SHA1
SRTCP authentication key length	160 bits
SRTCP authentication tag length	80 bits

Table 7: The ARIA_256_CTR_HMAC_SHA1_32 Crypto Suite

Parameter	Value
Master key length	128 bits
Master salt length	96 bits
Key Derivation Function	ARIA_128_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	AEAD_ARIA_128_GCM
AEAD authentication tag length	128 bits

Table 8: The AEAD_ARIA_128_GCM Crypto Suite

Parameter	Value
Master key length	256 bits
Master salt length	96 bits
Key Derivation Function	ARIA_256_CTR_PRF
Default key lifetime (SRTP)	2^48 packets
Default key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	AEAD_ARIA_256_GCM
AEAD authentication tag length	128 bits

Table 9: The AEAD_ARIA_256_GCM Crypto Suite

3. IANA Considerations

SDP Security Descriptions [RFC4568] defines SRTP "crypto suites". In order to allow SDP to signal the use of the algorithms defined in this document, IANA is requested to add the below crypto suites to the "SRTP Crypto Suite Registrations" created by [RFC4568], at time of writing located on the following IANA page: http://www.iana.org/assignments/sdp-security-descriptions/ .

```
        srtp-crypto-suite-ext = "ARIA_128_CTR_HMAC_SHA1_80"/
                                "ARIA_128_CTR_HMAC_SHA1_32"/
                                "ARIA_192_CTR_HMAC_SHA1_80"/
                                "ARIA_192_CTR_HMAC_SHA1_32"/
                                "ARIA_256_CTR_HMAC_SHA1_80"/
                                "ARIA_256_CTR_HMAC_SHA1_32"/
                                "AEAD_ARIA_128_GCM"        /
                                "AEAD_ARIA_256_GCM"        /
                                srtp-crypto-suite-ext
```

4. References

4.1. Normative References

[I-D.ietf-avtcore-aria-srtp]
Kim, W., Lee, J., Park, J., and D. Kwon, "The ARIA Algorithm and Its Use with the Secure Real-time Transport Protocol(SRTP)", draft-ietf-avtcore-aria-srtp-09 (work in progress), November 2015.

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

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <http://www.rfc-editor.org/info/rfc4568>.

4.2. Informative References

[ARIAKS] Korean Agency for Technology and Standards, "128 bit block encryption algorithm ARIA - Part 1: General (in Korean)", KS X 1213-1:2009, December 2009.

[ARIAPKCS]
RSA Laboratories, "Additional PKCS #11 Mechanisms", PKCS #11 v2.20 Amendment 3 Revision 1, January 2007.

[RFC5794] Lee, J., Lee, J., Kim, J., Kwon, D., and C. Kim, "A Description of the ARIA Encryption Algorithm", RFC 5794, DOI 10.17487/RFC5794, March 2010, <http://www.rfc-editor.org/info/rfc5794>.

Authors' Addresses

Woo-Hwan Kim
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: whkim5@nsr.re.kr

Jungkeun Lee
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: jklee@nsr.re.kr

Je Hong Park
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: jhpark@nsr.re.kr

Daesung Kwon
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: ds_kwon@nsr.re.kr

Dong-Chan Kim
Kookmin University
77 Jeongneung-ro, Seongbuk-gu
Seoul 02707
Korea

EMail: dckim@kookmin.ac.kr

AVTCore
Internet-Draft
Intended status: Informational
Expires: May 29, 2016

W. Kim
J. Lee
J. Park
D. Kwon
NSRI

November 26, 2015

The ARIA Algorithm and Its Use with the Secure Real-time Transport Protocol(SRTP)
draft-ietf-avtcore-aria-srtp-09

Abstract

This document defines the use of the ARIA block cipher algorithm within the Secure Real-time Transport Protocol (SRTP). It details two modes of operation (CTR, GCM) and a SRTP Key Derivation Function for ARIA. Additionally, this document defines DTLS-SRTP protection profiles and MIKEY parameter sets for the use with ARIA.

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 29, 2016.

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

Table of Contents

1. Introduction

This document defines the use of the ARIA [RFC5794] block cipher algorithm in the Secure Real-time Transport Protocol (SRTP) [RFC3711] for providing confidentiality for the Real-time Transport Protocol (RTP) [RFC3550] traffic and for the RTP Control Protocol (RTCP) [RFC3550] traffic.

1.1. ARIA

ARIA is a general-purpose block cipher algorithm developed by Korean cryptographers in 2003. It is an iterated block cipher with 128-, 192-, and 256-bit keys and encrypts 128-bit blocks in 12, 14, and 16

rounds, depending on the key size. It is secure and suitable for most software and hardware implementations on 32-bit and 8-bit processors. It was established as a Korean standard block cipher algorithm in 2004 [ARIAKS] and has been widely used in Korea, especially for government-to-public services. It was included in PKCS #11 in 2007 [ARIAPKCS]. The algorithm specification and object identifiers are described in [RFC5794].

1.2. Terminology

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

2. Cryptographic Transforms

Block ciphers ARIA and AES share common characteristics including mode, key size, and block size. ARIA does not have any restrictions for modes of operation that are used with this block cipher. We define two modes of running ARIA within the SRTP protocol, (1) ARIA in Counter Mode (ARIA-CTR) and (2) ARIA in Galois/Counter Mode (ARIA-GCM).

2.1. ARIA-CTR

Section 4.1.1 of [RFC3711] defines AES-128 counter mode encryption, which it refers to as "AES_CM". Section 2 of [RFC6188] defines "AES_192_CM" and "AES_256_CM" in SRTP. ARIA counter modes are defined in the same manner except that each invocation of AES is replaced by that of ARIA [RFC5794], and are denoted by ARIA_128_CTR, ARIA_192_CTR and ARIA_256_CTR respectively, according to the key lengths. The plaintext inputs to the block cipher are formed as in AES-CTR(AES_CM, AES_192_CM, AES_256_CM) and the block cipher outputs are processed as in AES-CTR. Note that, ARIA-CTR MUST be used only in conjunction with an authentication transform.

Section 3.2 of [RFC6904] defines AES-CTR for SRTP header extension keystream generation. When ARIA-CTR is used, the header extension keystream SHALL be generated in the same manner except that each invocation of AES is replaced by that of ARIA [RFC5794].

2.2. ARIA-GCM

GCM (Galois Counter Mode) [GCM][RFC5116] is an AEAD (Authenticated Encryption with Associated Data) block cipher mode. A detailed description of ARIA-GCM is defined similarly as AES-GCM found in [RFC5116][RFC5282].

The document [I-D.ietf-avtcore-srtp-aes-gcm] describes the use of AES-GCM with SRTP [RFC3711][RFC6904]. The use of ARIA-GCM with SRTP is defined the same as that of AES-GCM except that each invocation of AES is replaced by ARIA [RFC5794]. When [RFC6904] is in use, a separate keystream to encrypt selected RTP header extension elements MUST be generated in the same manner defined in [I-D.ietf-avtcore-srtp-aes-gcm] except that AES-CTR is replaced by ARIA-CTR.

3. Key Derivation Functions

Section 4.3.3 of [RFC3711] defines the AES-128 counter mode key derivation function, which it refers to as "AES-CM PRF". Section 3 of [RFC6188] defines the AES-192 counter mode key derivation function and the AES-256 counter mode key derivation function, which it refers to as "AES_192_CM_PRF" and "AES_256_CM_PRF" respectively. The ARIA-CTR PRF is defined in a same manner except that each invocation of AES replaced by that of ARIA. According to the key lengths of underlying encryption algorithm, ARIA-CTR PRFs are denoted by "ARIA_128_CTR_PRF", "ARIA_192_CTR_PRF" and "ARIA_256_CTR_PRF". The usage requirements of [RFC6188][I-D.ietf-avtcore-srtp-aes-gcm] regarding the AES-CM PRF apply to the ARIA-CTR PRF as well.

4. Protection Profiles

This section defines SRTP Protection Profiles that use the ARIA transforms and key derivation functions defined in this document. The following list indicates the SRTP transform parameters for each protection profile. Those are described for use with DTLS-SRTP [RFC5764].

The parameters cipher_key_length, cipher_salt_length, auth_key_length, and auth_tag_length express the number of bits in the values to which they refer. The maximum_lifetime parameter indicates the maximum number of packets that can be protected with each single set of keys when the parameter profile is in use. All of these parameters apply to both RTP and RTCP, unless the RTCP parameters are separately specified.

SRTP_ARIA_128_CTR_HMAC_SHA1_80

cipher:	ARIA_128_CTR
cipher_key_length:	128 bits
cipher_salt_length:	112 bits
key derivation function:	ARIA_128_CTR_PRF
auth_function:	HMAC-SHA1
auth_key_length:	160 bits
auth_tag_length:	80 bits
maximum_lifetime:	at most 2^31 SRTCP packets and

```
                                  at most 2^48 SRTP packets

   SRTP_ARIA_128_CTR_HMAC_SHA1_32
           cipher:                 ARIA_128_CTR
           cipher_key_length:      128 bits
           cipher_salt_length:     112 bits
           key derivation function: ARIA_128_CTR_PRF
           auth_function:          HMAC-SHA1
           auth_key_length:        160 bits
           SRTP auth_tag_length:   32 bits
           SRTCP auth_tag_length:  80 bits
           maximum_lifetime:       at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_ARIA_192_CTR_HMAC_SHA1_80
           cipher:                 ARIA_192_CTR
           cipher_key_length:      192 bits
           cipher_salt_length:     112 bits
           key derivation function: ARIA_192_CTR_PRF
           auth_function:          HMAC-SHA1
           auth_key_length:        160 bits
           auth_tag_length:        80 bits
           maximum_lifetime:       at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_ARIA_192_CTR_HMAC_SHA1_32
           cipher:                 ARIA_192_CTR
           cipher_key_length:      192 bits
           cipher_salt_length:     112 bits
           key derivation function: ARIA_192_CTR_PRF
           auth_function:          HMAC-SHA1
           auth_key_length:        160 bits
           SRTP auth_tag_length:   32 bits
           SRTCP auth_tag_length:  80 bits
           maximum_lifetime:       at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_ARIA_256_CTR_HMAC_SHA1_80
           cipher:                 ARIA_256_CTR
           cipher_key_length:      256 bits
           cipher_salt_length:     112 bits
           key derivation function: ARIA_256_CTR_PRF
           auth_function:          HMAC-SHA1
           auth_key_length:        160 bits
           auth_tag_length:        80 bits
           maximum_lifetime:       at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets
```

```
SRTP_ARIA_256_CTR_HMAC_SHA1_32
        cipher:                   ARIA_256_CTR
        cipher_key_length:        256 bits
        cipher_salt_length:       112 bits
        key derivation function:  ARIA_256_CTR_PRF
        auth_function:            HMAC-SHA1
        auth_key_length:          160 bits
        SRTP auth_tag_length:     32 bits
        SRTCP auth_tag_length:    80 bits
        maximum_lifetime:         at most 2^31 SRTCP packets and
                                  at most 2^48 SRTP packets

SRTP_AEAD_ARIA_128_GCM
        cipher:                   ARIA_128_GCM
        cipher_key_length:        128 bits
        cipher_salt_length:       96 bits
        aead_auth_tag_length:     128 bits
        auth_function:            NULL
        auth_key_length:          N/A
        auth_tag_length:          N/A
        key derivation function:  ARIA_128_CTR_PRF
        maximum_lifetime:         at most 2^31 SRTCP packets and
                                  at most 2^48 SRTP packets

SRTP_AEAD_ARIA_256_GCM
        cipher:                   ARIA_256_GCM
        cipher_key_length:        256 bits
        cipher_salt_length:       96 bits
        aead_auth_tag_length:     128 bits
        auth_function:            NULL
        auth_key_length:          N/A
        auth_tag_length:          N/A
        key derivation function:  ARIA_256_CTR_PRF
        maximum_lifetime:         at most 2^31 SRTCP packets and
                                  at most 2^48 SRTP packets
```

The ARIA-CTR protection profiles use the same authentication transform that is mandatory to implement in SRTP, HMAC-SHA1 with a 160-bit key.

Note that SRTP Protection Profiles which use AEAD algorithms do not specify an auth_function, auth_key_length, or auth_tag_length, since they do not use a separate auth_function, auth_key, or auth_tag. The term aead_auth_tag_length is used to emphasize that this refers to the authentication tag provided by the AEAD algorithm and that this tag is not located in the authentication tag field provided by SRTP/SRTCP.

The PRFs for ARIA protect profiles with SRTP are defined by ARIA-CTR PRF of the equal key length with the encryption algorithm (see Section 2). SRTP_ARIA_128_CTR_HMAC and SRTP_AEAD_ARIA_128_GCM MUST use the ARIA_128_CTR_PRF Key Derivation Function. SRTP_ARIA_192_CTR_HMAC MUST use that ARIA_192_CTR_PRF Key Derivation Function. And SRTP_ARIA_256_CTR_HMAC and SRTP_AEAD_ARIA_256_GCM MUST use the ARIA_256_CTR_PRF Key Derivation Function.

MIKEY specifies the SRTP protection profile definition separately from the key length (which is specified by the Session Encryption key length) and the authentication tag length. The DTLS-SRTP [RFC5764] protection profiles are mapped to MIKEY parameter sets as shown below.

	Encryption Algorithm	Encryption Key Length	Auth. Tag Length
SRTP_ARIA_128_CTR_HMAC_80	ARIA-CTR	16 octets	10 octets
SRTP_ARIA_128_CTR_HMAC_32	ARIA-CTR	16 octets	4 octets
SRTP_ARIA_192_CTR_HMAC_80	ARIA-CTR	24 octets	10 octets
SRTP_ARIA_192_CTR_HMAC_32	ARIA-CTR	24 octets	4 octets
SRTP_ARIA_256_CTR_HMAC_80	ARIA-CTR	32 octets	10 octets
SRTP_ARIA_256_CTR_HMAC_32	ARIA-CTR	32 octets	4 octets

Figure 1: Mapping MIKEY parameters to ARIA-CTR with HMAC algorithm

	Encryption Algorithm	Encryption Key Length	AEAD Auth. Tag Length
SRTP_AEAD_ARIA_128_GCM	ARIA-GCM	16 octets	16 octets
SRTP_AEAD_ARIA_256_GCM	ARIA-GCM	32 octets	16 octets

Figure 2: Mapping MIKEY parameters to AEAD algorithm

5. Security Considerations

At the time of writing this document no security problem has been found on ARIA (see [TSL]).

The security considerations in [GCM] [RFC3711] [RFC5116] [RFC6188] [RFC6904] [I-D.ietf-avtcore-srtp-aes-gcm] apply to this document as well. Protection profiles with short tag length may be considered for specific application environments stated in Section 7.5 of

[RFC3711], but the risk of weak authentication described in Section 9.5.1 of [RFC3711] should be taken into account.

6. IANA Considerations

6.1. DTLS-SRTP

DTLS-SRTP [RFC5764] defines a DTLS-SRTP "SRTP Protection Profile". In order to allow the use of the algorithms defined in this document in DTLS-SRTP, IANA is requested to add the below protection profiles to the "DTLS-SRTP Protection Profiles" created by [RFC5764], at time of writing located on the following IANA page: http://www.iana.org/assignments/srtp-protection/ .

```
SRTP_ARIA_128_CTR_HMAC_SHA1_80 = {TBD,TBD}
SRTP_ARIA_128_CTR_HMAC_SHA1_32 = {TBD,TBD}
SRTP_ARIA_192_CTR_HMAC_SHA1_80 = {TBD,TBD}
SRTP_ARIA_192_CTR_HMAC_SHA1_32 = {TBD,TBD}
SRTP_ARIA_256_CTR_HMAC_SHA1_80 = {TBD,TBD}
SRTP_ARIA_256_CTR_HMAC_SHA1_32 = {TBD,TBD}
SRTP_AEAD_ARIA_128_GCM = {TBD,TBD}
SRTP_AEAD_ARIA_256_GCM = {TBD,TBD}
```

6.2. MIKEY

[RFC3830] and [RFC5748] define encryption algorithms and PRFs for the SRTP policy in MIKEY. In order to allow the use of the algorithms defined in this document in MIKEY, IANA is requested to add the below three encryption algorithms to the "MIKEY Security Protocol Parameters SRTP Type 0 (Encryption algorithm)" and to add the below PRF to the "MIKEY Security Protocol Parameters SRTP Type 5 (Pseudo Random Function)" created by [RFC3830], at time of writing located on the following IANA page: http://www.iana.org/assignments/mikey-payloads/ .

SRTP Enc. alg	Value
ARIA-CTR	TBD
ARIA-GCM	TBD

Default session encryption key length is 16 octets.

SRTP PRF	Value
ARIA-CTR	TBD

7. References

7.1. Normative References

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, November 2007.

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

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

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

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

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, DOI 10.17487/RFC3830, August 2004, <http://www.rfc-editor.org/info/rfc3830>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <http://www.rfc-editor.org/info/rfc5116>.

[RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, DOI 10.17487/RFC5282, August 2008, <http://www.rfc-editor.org/info/rfc5282>.

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

[RFC6188] McGrew, D., "The Use of AES-192 and AES-256 in Secure RTP", RFC 6188, DOI 10.17487/RFC6188, March 2011, <http://www.rfc-editor.org/info/rfc6188>.

[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>.

7.2. Informative References

[ARIAKS] Korean Agency for Technology and Standards, "128 bit block encryption algorithm ARIA - Part 1: General (in Korean)", KS X 1213-1:2009, December 2009.

[ARIAPKCS] RSA Laboratories, "Additional PKCS #11 Mechanisms", PKCS #11 v2.20 Amendment 3 Revision 1, January 2007.

[RFC5748] Yoon, S., Jeong, J., Kim, H., Jeong, H., and Y. Won, "IANA Registry Update for Support of the SEED Cipher Algorithm in Multimedia Internet KEYing (MIKEY)", RFC 5748, DOI 10.17487/RFC5748, August 2010, <http://www.rfc-editor.org/info/rfc5748>.

[RFC5794] Lee, J., Lee, J., Kim, J., Kwon, D., and C. Kim, "A Description of the ARIA Encryption Algorithm", RFC 5794, DOI 10.17487/RFC5794, March 2010, <http://www.rfc-editor.org/info/rfc5794>.

[TSL] Tang, X., Sun, B., Li, R., Li, C., and J. Yin, "A meet-in-the-middle attack on reduced-round ARIA", The Journal of Systems and Software Vol.84(10), pp. 1685-1692, October 2011.

Appendix A. Test Vectors

All values are in hexadecimal and represented by the network order (called big endian).

A.1. ARIA-CTR Test Vectors

Common values are organized as follows:

Rollover Counter:	00000000
Sequence Number:	315e
SSRC:	20e8f5eb
Authentication Key:	f93563311b354748c978913795530631 16452309
Session Salt:	cd3a7c42c671e0067a2a2639b43a
Initialization Vector:	cd3a7c42e69915ed7a2a263985640000
RTP header:	8008315ebf2e6fe020e8f5eb
RTP Payload:	f57af5fd4ae19562976ec57a5a7ad55a 5af5c5e5c5fdf5c55ad57a4a7272d572 62e9729566ed66e97ac54a4a5a7ad5e1 5ae5fdd5fd5ac5d56ae56ad5c572d54a e54ac55a956afd6aed5a4ac562957a95 16991691d572fd14e97ae962ed7a9f4a 955af572e162f57a956666e17ae1f54a 95f566d54a66e16e4afd6a9f7ae1c5c5 5ae5d56afde916c5e94a6ec56695e14a fde1148416e94ad57ac5146ed59d1cc5

A.1.1. SRTP_ARIA_128_CTR_HMAC_SHA1_80

Session Key: 0c5ffd37a11edc42c325287fc0604f2e

Encrypted RTP Payload: 1bf753f412e6f35058cc398dc851aae3
a6ccdcb463fbed9cfb3de2fb76fdffa9
e481f5efb64c92487f59dabbc7cc72da
092485f3fbad87888820b86037311fa4
4330e18a59a1e1338ba2c21458493a57
463475c54691f91cec785429119e0dfc
d9048f90e07fecd50b528e8c62ee6e71
445de5d7f659405135aff3604c2ca4ff
4aaca40809cb9eee42cc4ad232307570
81ca289f2851d3315e9568b501fdce6d

Authenticated portion || Rollover Counter:
8008315ebf2e6fe020e8f5eb1bf753f4
12e6f35058cc398dc851aae3a6ccdcb4
63fbed9cfb3de2fb76fdffa9e481f5ef
b64c92487f59dabbc7cc72da092485f3
fbad87888820b86037311fa44330e18a
59a1e1338ba2c21458493a57463475c5
4691f91cec785429119e0dfcd9048f90
e07fecd50b528e8c62ee6e71445de5d7
f659405135aff3604c2ca4ff4aaca408
09cb9eee42cc4ad23230757081ca289f
2851d3315e9568b501fdce6d00000000

Authentication Tag: f9de4e729054672b0e35

A.1.2. SRTP_ARIA_192_CTR_HMAC_SHA1_80

Session Key: 0c5ffd37a11edc42c325287fc0604f2e
3e8cd5671a00fe32

Encrypted RTP Payload: 86f4556486642caa67e9b40fef2acda0
6d442517d8d58c15e3e0b5c13a78b8b2
838b7b96961e11acb2af81348272888c
fd9d168ba091fe3e4f7f83c7871570a9
aa9f995036e44c35cb742b601e8d8d08
48320bad732929103f1bfbb1ae873178
0479c5df2d4d41f78f6b96d6832db3db
6af8b3612b27e18a0a29a8a1d280437e
b8dad58e78658ec3b069d7329431c356
c5e612b3dde5bd3f6c9f42f39cf35d3a

Authenticated portion || Rollover Counter:
8008315ebf2e6fe020e8f5eb86f45564
86642caa67e9b40fef2acda06d442517
d8d58c15e3e0b5c13a78b8b2838b7b96
961e11acb2af81348272888cfd9d168b
a091fe3e4f7f83c7871570a9aa9f9950
36e44c35cb742b601e8d8d0848320bad
732929103f1bfbb1ae8731780479c5df
2d4d41f78f6b96d6832db3db6af8b361
2b27e18a0a29a8a1d280437eb8dad58e
78658ec3b069d7329431c356c5e612b3
dde5bd3f6c9f42f39cf35d3a00000000

Authentication Tag: 3935fa37ee96dbc550d5

A.1.3. SRTP_ARIA_256_CTR_HMAC_SHA1_80

Session Key: 0c5ffd37a11edc42c325287fc0604f2e
3e8cd5671a00fe3216aa5eb105783b54

Encrypted RTP Payload: c424c59fd5696305e5b13d8e8ca76566
17ccd7471088af9debf07b55c750f804
a5ac2b737be48140958a9b420524112a
e72e4da5bca59d2b1019ddd7dbdc30b4
3d5f046152ced40947d62d2c93e7b8e5
0f02db2b6b61b010e4c1566884de1fa9
702cdf8157e8aedfe3dd77c76bb50c25
ae4d624615c15acfdeeb5f79482aaa01
d3e4c05eb601eca2bd10518e9d46b021
16359232e9eac0fabd05235dd09e6dea

Authenticated portion || Rollover Counter:
8008315ebf2e6fe020e8f5ebc424c59f
d5696305e5b13d8e8ca7656617ccd747
1088af9debf07b55c750f804a5ac2b73
7be48140958a9b420524112ae72e4da5
bca59d2b1019ddd7dbdc30b43d5f0461
52ced40947d62d2c93e7b8e50f02db2b
6b61b010e4c1566884de1fa9702cdf81
57e8aedfe3dd77c76bb50c25ae4d6246
15c15acfdeeb5f79482aaa01d3e4c05e
b601eca2bd10518e9d46b02116359232
e9eac0fabd05235dd09e6dea00000000

Authentication Tag: 192f515fab04bbb4e62c

A.2. ARIA-GCM Test Vectors

Common values are organized as follows:

```
Rollover Counter:          00000000
Sequence Number:           315e
SSRC:                      20e8f5eb
Encryption Salt:           00000000000000000000000000

Initialization Vector:     000020e8f5eb00000000315e
RTP Payload:               f57af5fd4ae19562976ec57a5a7ad55a
                           5af5c5e5c5fdf5c55ad57a4a7272d572
                           62e9729566ed66e97ac54a4a5a7ad5e1
                           5ae5fdd5fd5ac5d56ae56ad5c572d54a
                           e54ac55a956afd6aed5a4ac562957a95
                           16991691d572fd14e97ae962ed7a9f4a
                           955af572e162f57a956666e17ae1f54a
                           95f566d54a66e16e4afd6a9f7ae1c5c5
                           5ae5d56afde916c5e94a6ec56695e14a
                           fde1148416e94ad57ac5146ed59d1cc5
Associated Data:           8008315ebf2e6fe020e8f5eb
```

The length of encrypted payload is larger than that of payload by 16 octets which the length of the tag from GCM.

A.2.1. SRTP_AEAD_ARIA_128_GCM

```
Key:                       e91e5e75da65554a48181f3846349562

Encrypted RTP Payload:     4d8a9a0675550c704b17d8c9ddc81a5c
                           d6f7da34f2fe1b3db7cb3dfb9697102e
                           a0f3c1fc2dbc873d44bceeae8e444297
                           4ba21ff6789d3272613fb9631a7cf3f1
                           4bacbeb421633a90ffbe58c2fa6bdca5
                           34f10d0de0502ce1d531b6336e588782
                           78531e5c22bc6c85bbd784d78d9e680a
                           a19031aaf89101d669d7a3965c1f7e16
                           229d7463e0535f4e253f5d18187d40b8
                           ae0f564bd970b5e7e2adfb211e89a953
                           5abace3f37f5a736f4be984bbffbedc1
```

A.2.2. SRTP_AEAD_ARIA_256_GCM

```
Key:                        0c5ffd37a11edc42c325287fc0604f2e
                            3e8cd5671a00fe3216aa5eb105783b54

Encrypted RTP Payload:      6f9e4bcbc8c85fc0128fb1e4a0a20cb9
                            932ff74581f54fc013dd054b19f99371
                            425b352d97d3f337b90b63d1b082adee
                            ea9d2d7391897d591b985e55fb50cb53
                            50cf7d38dc27dda127c078a149c8eb98
                            083d66363a46e3726af217d3a00275ad
                            5bf772c7610ea4c23006878f0ee69a83
                            97703169a419303f40b72e4573714d19
                            e2697df61e7c7252e5abc6bade876ac4
                            961bfac4d5e867afca351a48aed52822
                            e210d6ced2cf430ff841472915e7ef48
```

A.3. Key Derivation Test Vector

This section provides test vectors for the default key derivation function, which uses ARIA in Counter Mode. In the following, we walk through the initial key derivation for the ARIA Counter Mode cipher, which requires a 16/24/32 octet session encryption key according to the session encryption key length and a 14 octet session salt, and an authentication function which requires a 94 octet session authentication key. These values are called the cipher key, the cipher salt, and the auth key in the following. The test vectors are generated in the same way with the test vectors of key derivation functions in [RFC3711] and [RFC6188] but with each invocation of AES replaced with an invocation of ARIA.

A.3.1. ARIA_128_CTR_PRF

The inputs to the key derivation function are the 16 octet master key and the 14 octet master salt:

```
master key:  e1f97a0d3e018be0d64fa32c06de4139
master salt: 0ec675ad498afeebb6960b3aabe6

index DIV kdr:                 000000000000
label:                       00
master salt:   0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:           0ec675ad498afeebb6960b3aabe6     (x, PRF input)

x*2^16:        0ec675ad498afeebb6960b3aabe60000 (ARIA-CTR input)

cipher key:    dbd85a3c4d9219b3e81f7d942e299de4 (ARIA-CTR output)
```

ARIA-CTR protection profile requires 14 octet cipher salt while ARIA-GCM protection profile requires 12 octet cipher salt.

```
  index DIV kdr:                000000000000
  label:                      02
  master salt:   0ec675ad498afeebb6960b3aabe6
  ---------------------------------------------
  xor:           0ec675ad498afee9b6960b3aabe6     (x, PRF input)

  x*2^16:        0ec675ad498afee9b6960b3aabe60000 (ARIA-CTR input)

                 9700657f5f34161830d7d85f5dc8be7f (ARIA-CTR output)

  cipher salt:   9700657f5f34161830d7d85f5dc8     (ARIA-CTR profile)
                 9700657f5f34161830d7d85f         (ARIA-GCM profile)
  index DIV kdr:                000000000000
  label:                      01
  master salt:   0ec675ad498afeebb6960b3aabe6
  ---------------------------------------------
  xor:           0ec675ad498afeeab6960b3aabe6     (x, PRF input)

  x*2^16:        0ec675ad498afeeab6960b3aabe60000 (ARIA-CTR input)
```

Below, the auth key is shown on the left, while the corresponding ARIA input blocks are shown on the right.

```
  auth key                          ARIA input blocks

  d021877bd3eaf92d581ed70ddc050e03  0ec675ad498afeeab6960b3aabe60000
  f11257032676f2a29f57b21abd3a1423  0ec675ad498afeeab6960b3aabe60001
  769749bdc5dd9ca5b43ca6b6c1f3a7de  0ec675ad498afeeab6960b3aabe60002
  4047904bcf811f601cc03eaa5d7af6db  0ec675ad498afeeab6960b3aabe60003
  9f88efa2e51ca832fc2a15b126fa7be2  0ec675ad498afeeab6960b3aabe60004
  469af896acb1852c31d822c45799      0ec675ad498afeeab6960b3aabe60005
```

A.3.2. ARIA_192_CTR_PRF

The inputs to the key derivation function are the 24 octet master key and the 14 octet master salt:

```
master key:  0c5ffd37a11edc42c325287fc0604f2e3e8cd5671a00fe32
master salt: 0ec675ad498afeebb6960b3aabe6

index DIV kdr:              000000000000
label:                    00
master salt: 0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:         0ec675ad498afeebb6960b3aabe6     (x, PRF input)

x*2^16:      0ec675ad498afeebb6960b3aabe60000 (ARIA-CTR input)

cipher key:  f320af2386a1cde64c3aa5f55d68002e (ARIA-CTR 1st output)
             d13cbe548b627649                 (ARIA-CTR 2nd Output)
```

ARIA-CTR protection profile requires 14 octet cipher salt.

```
index DIV kdr:               000000000000
label:                     02
master salt:  0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:          0ec675ad498afee9b6960b3aabe6     (x, PRF input)

x*2^16:       0ec675ad498afee9b6960b3aabe60000 (ARIA-CTR input)

              55c7e3555baf0fdc91c589cfb871b098 (ARIA-CTR output)

cipher salt:  55c7e3555baf0fdc91c589cfb871     (ARIA-CTR profile)

index DIV kdr:               000000000000
label:                     01
master salt:  0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:          0ec675ad498afeeab6960b3aabe6     (x, PRF input)

x*2^16:       0ec675ad498afeeab6960b3aabe60000 (ARIA-CTR input)
```

Below, the auth key is shown on the left, while the corresponding ARIA input blocks are shown on the right.

```
 auth key                          ARIA input blocks

116902524517f7e767a979ad7678d53a   0ec675ad498afeeab6960b3aabe60000
8cae05a5c9a315d1304f634c81a06617   0ec675ad498afeeab6960b3aabe60001
31fe099d4dcd2202421fe01fc12c65ad   0ec675ad498afeeab6960b3aabe60002
009e920031654855af5d9e820a7831e0   0ec675ad498afeeab6960b3aabe60003
bc2b4744d2a33053eb685138252f2d82   0ec675ad498afeeab6960b3aabe60004
9a89f4a9aa4f97fde0cce9bad3d5       0ec675ad498afeeab6960b3aabe60005
```

A.3.3. ARIA_256_CTR_PRF

The inputs to the key derivation function are the 32 octet master key and the 14 octet master salt:

```
master key:  0c5ffd37a11edc42c325287fc0604f2e
             3e8cd5671a00fe3216aa5eb105783b54
master salt: 0ec675ad498afeebb6960b3aabe6

index DIV kdr:              000000000000
label:                    00
master salt: 0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:         0ec675ad498afeebb6960b3aabe6     (x, PRF input)

x*2^16:      0ec675ad498afeebb6960b3aabe60000 (ARIA-CTR input)

cipher key:  0649a09d93755fe9c2b2efba1cce930a (ARIA-CTR 1st output)
             f2e76ce8b77e4b175950321aa94b0cf4 (ARIA-CTR 2nd output)
```

ARIA-CTR protection profile requires 14 octet cipher salt while ARIA-GCM protection profile requires 12 octet cipher salt.

```
index DIV kdr:               000000000000
label:                     02
master salt:  0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:          0ec675ad498afee9b6960b3aabe6     (x, PRF input)

x*2^16:       0ec675ad498afee9b6960b3aabe60000 (ARIA-CTR input)

              194abaa8553a8eba8a413a340fc80a3d (ARIA-CTR output)

cipher salt:  194abaa8553a8eba8a413a340fc8     (ARIA-CTR profile)
              194abaa8553a8eba8a413a34         (ARIA-GCM profile)

index DIV kdr:               000000000000
label:                     01
master salt:  0ec675ad498afeebb6960b3aabe6
-----------------------------------------------
xor:          0ec675ad498afeeab6960b3aabe6     (x, PRF input)

x*2^16:       0ec675ad498afeeab6960b3aabe60000 (ARIA-CTR input)
```

Below, the auth key is shown on the left, while the corresponding ARIA input blocks are shown on the right.

auth key	ARIA input blocks
e58d42915873b71899234807334658f2	0ec675ad498afeeab6960b3aabe60000
0bc460181d06e02b7a9e60f02ff10bfc	0ec675ad498afeeab6960b3aabe60001
9ade3795cf78f3e0f2556d9d913470c4	0ec675ad498afeeab6960b3aabe60002
e82e45d254bfb8e2933851a3930ffe7d	0ec675ad498afeeab6960b3aabe60003
fca751c03ec1e77e35e28dac4f17d1a5	0ec675ad498afeeab6960b3aabe60004
80bdac028766d3b1e8f5a41faa3c	0ec675ad498afeeab6960b3aabe60005

Authors' Addresses

Woo-Hwan Kim
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 305-350
Korea

EMail: whkim5@ensec.re.kr

Jungkeun Lee
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 305-350
Korea

EMail: jklee@ensec.re.kr

Je-Hong Park
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 305-350
Korea

EMail: jhpark@ensec.re.kr

Daesung Kwon
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 305-350
Korea

EMail: ds_kwon@ensec.re.kr

AVTCore
Internet-Draft
Obsoletes: 5285 (if approved)
Intended status: Standards Track
Expires: December 5, 2017

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

June 3, 2017

A General Mechanism for RTP Header Extensions
draft-ietf-avtcore-rfc5285-bis-12.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 December 5, 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-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 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 dp 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.

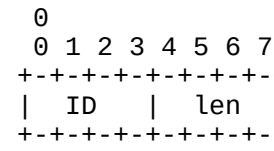
Note carefully that the one-byte header form allows for data lengths between 1 and 16 bytes, by adding 1 to the signaled length value (thus, 0 in the length field indicates 1 byte of data follows). This allows for the important case of 16-byte payloads. This addition is not performed for the two-byte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

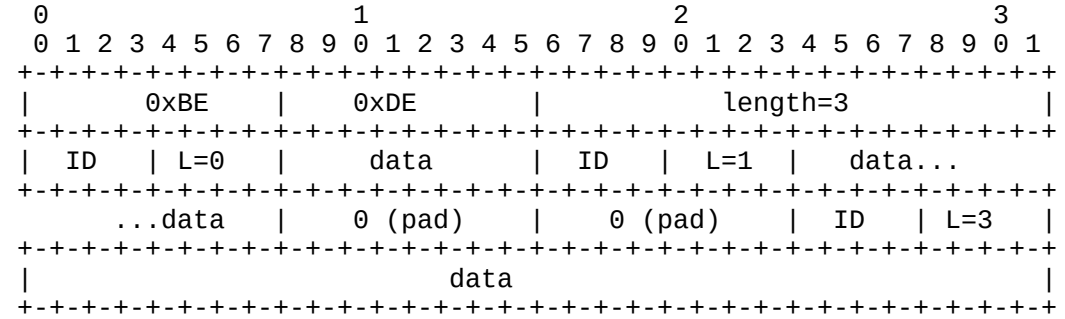


The 4-bit ID is the local identifier of this element in the range 1-14 inclusive. In the signaling section, this is referred to as the valid range.

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.

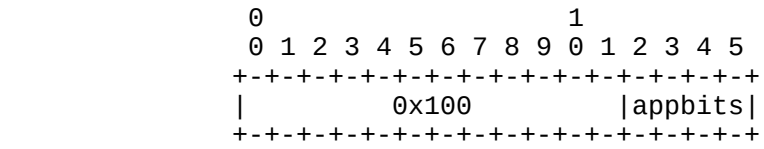
The 4-bit length is the number minus one of data bytes of this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows, and a value of 15 (the maximum) indicates element data of 16 bytes. (This permits carriage of 16-byte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



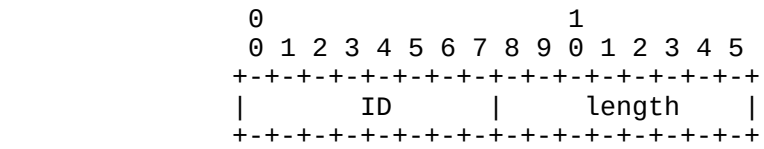
4.3. Two-Byte Header

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.

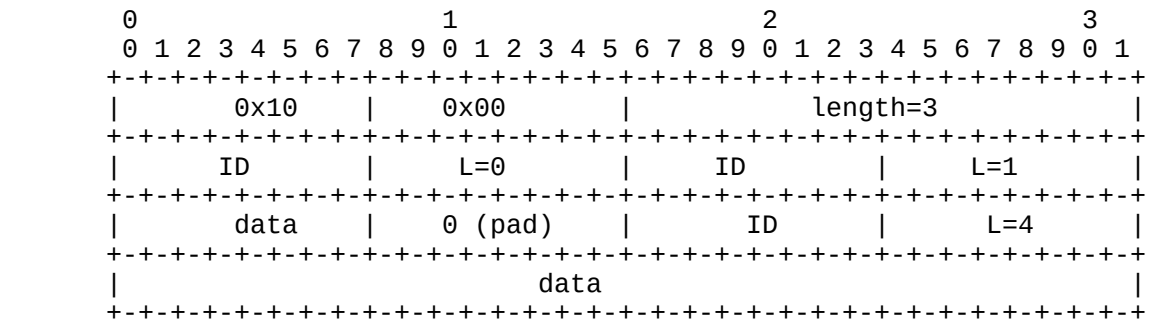
Each extension element starts with a byte containing an ID and a byte containing a length:



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 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 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:

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 normal 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.

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 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 do 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. It can be noted that the default SRTP ciphers (AES CM 128 bits with HMAC-SHA1) are relative weak and more modern ciphers are stronger and should be considered.

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 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 registeration 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].

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[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>.

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: Standards Track
Expires: September 14, 2017

E. Berger
S. Nandakumar
M. Zanaty
Cisco Systems

March 13, 2017

Frame Marking RTP Header Extension
draft-ietf-avtext-framemarking-04

Abstract

This document describes a Frame Marking RTP header extension used to convey information about video frames that is critical for error recovery and packet forwarding in RTP middleboxes or network nodes. It is most useful when media is encrypted, and essential when the middlebox or node has no access to the media decryption keys. It is also useful for codec-agnostic processing of encrypted or unencrypted media, while it also supports extensions for codec-specific information.

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.

Copyright Notice

Table of Contents

1. Introduction

Many widely deployed RTP [RFC3550] topologies [RFC7667] used in modern voice and video conferencing systems include a centralized component that acts as an RTP switch. It receives voice and video streams from each participant, which may be encrypted using SRTP [RFC3711], or extensions that provide participants with private media via end-to-end encryption where the switch has no access to media decryption keys. The goal is to provide a set of streams back to the participants which enable them to render the right media content. In a simple video configuration, for example, the goal will be that each participant sees and hears just the active speaker. In that case, the goal of the switch is to receive the voice and video streams from each participant, determine the active speaker based on energy in the voice packets, possibly using the client-to-mixer audio level RTP header extension [RFC6464], and select the corresponding video stream for transmission to participants; see Figure 1.

In this document, an "RTP switch" is used as a common short term for the terms "switching RTP mixer", "source projecting middlebox", "source forwarding unit/middlebox" and "video switching MCU" as discussed in [RFC7667].

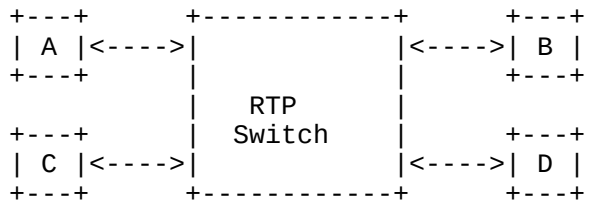


Figure 1: RTP switch

In order to properly support switching of video streams, the RTP switch typically needs some critical information about video frames in order to start and stop forwarding streams.

- Because of inter-frame dependencies, it should ideally switch video streams at a point where the first frame from the new speaker can be decoded by recipients without prior frames, e.g switch on an intra-frame.
- In many cases, the switch may need to drop frames in order to realize congestion control techniques, and needs to know which frames can be dropped with minimal impact to video quality.
- Furthermore, it is highly desirable to do this in a payload format-agnostic way which is not specific to each different video codec. Most modern video codecs share common concepts around frame types and other critical information to make this codec-agnostic handling possible.
- It is also desirable to be able to do this for SRTP without requiring the video switch to decrypt the packets. SRTP will encrypt the RTP payload format contents and consequently this data is not usable for the switching function without decryption, which may not even be possible in the case of end-to-end encryption of private media.

By providing meta-information about the RTP streams outside the encrypted media payload, an RTP switch can do codec-agnostic selective forwarding without decrypting the payload. This document specifies the necessary meta-information in an RTP header extension.

2. Key Words for Normative Requirements

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. Frame Marking RTP Header Extension

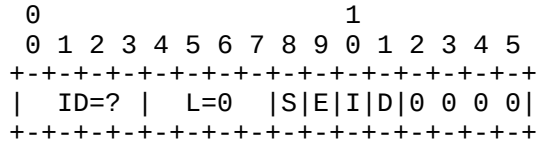
This specification uses RTP header extensions as defined in [RFC5285]. A subset of meta-information from the video stream is provided as an RTP header extension to allow an RTP switch to do generic selective forwarding of video streams encoded with potentially different video codecs.

The Frame Marking RTP header extension is encoded using the one-byte header or two-byte header as described in [RFC5285]. The one-byte header format is used for examples in this memo. The two-byte header format is used when other two-byte header extensions are present in the same RTP packet, since mixing one-byte and two-byte extensions is not possible in the same RTP packet.

This extension is only specified for Source (not Redunadancy) RTP Streams [RFC7656] that carry video payloads. It is not specified for audio payloads, nor is it specified for Redundancy RTP Streams. The (separate) specifications for Redudancy RTP Streams often include provisions for recovering any header extensions that were part of the original source packet. Such provisions SHALL be followed to recover the Frame Marking RTP header extension of the original source packet.

3.1. Extension for Non-Scalable Streams

The following RTP header extension is RECOMMENDED for non-scalable streams. It MAY also be used for scalable streams if the sender has limited or no information about stream scalability. The ID is assigned per [RFC5285], and the length is encoded as L=0 which indicates 1 octet of data.

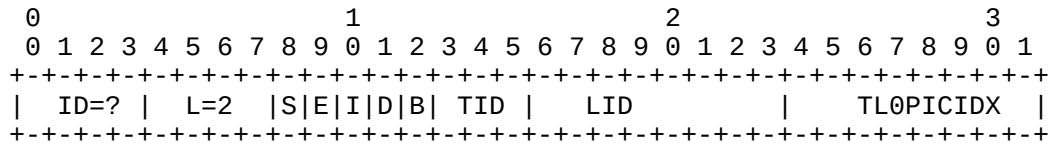


The following information are extracted from the media payload and sent in the Frame Marking RTP header extension.

- o S: Start of Frame (1 bit) - MUST be 1 in the first packet in a frame; otherwise MUST be 0.
- o E: End of Frame (1 bit) - MUST be 1 in the last packet in a frame; otherwise MUST be 0.
- o I: Independent Frame (1 bit) - MUST be 1 for frames that can be decoded independent of prior frames, e.g. intra-frame, VPX keyframe, H.264 IDR [RFC6184], H.265 IDR/CRA/BLA/RAP [RFC7798]; otherwise MUST be 0.
- o D: Discardable Frame (1 bit) - MUST be 1 for frames that can be discarded, and still provide a decodable media stream; otherwise MUST be 0.
- o The remaining (4 bits) - MUST be 0 for non-scalable streams.

3.2. Extension for Scalable Streams

The following RTP header extension is RECOMMENDED for scalable streams. It MAY also be used for non-scalable streams, in which case TID, LID and TL0PICIDX MUST be 0. The ID is assigned per [RFC5285], and the length is encoded as L=2 which indicates 3 octets of data.



The following information are extracted from the media payload and sent in the Frame Marking RTP header extension.

- o S: Start of Frame (1 bit) - MUST be 1 in the first packet in a frame within a layer; otherwise MUST be 0.
- o E: End of Frame (1 bit) - MUST be 1 in the last packet in a frame within a layer; otherwise MUST be 0.
- o I: Independent Frame (1 bit) - MUST be 1 for frames that can be decoded independent of prior frames, e.g. intra-frame, VPX keyframe, H.264 IDR [RFC6184], H.265 IDR/CRA/BLA/RAP [RFC7798]; otherwise MUST be 0.
- o D: Discardable Frame (1 bit) - MUST be 1 for frames that can be discarded, and still provide a decodable media stream; otherwise MUST be 0.
- o B: Base Layer Sync (1 bit) - MUST be 1 if this frame only depends on the base layer; otherwise MUST be 0. If no scalability is used, this MUST be 0.
- o TID: Temporal ID (3 bits) - The base temporal layer starts with 0, and increases with 1 for each higher temporal layer/sub-layer. If no scalability is used, this MUST be 0.

- o LID: Layer ID (8 bits) - Identifies the spatial and quality layer encoded. If no scalability is used, this MUST be 0 or omitted. When omitted, TL0PICIDX MUST also be omitted.
- o TL0PICIDX: Temporal Layer 0 Picture Index (8 bits) - Running index of base temporal layer 0 frames when TID is 0. When TID is not 0, this indicates a dependency on the given index. If no scalability is used, this MUST be 0 or omitted. When omitted, LID MUST also be omitted.

The layer information contained in TID and LID convey useful aspects of the layer structure that can be utilized in selective forwarding. Without further information about the layer structure, these identifiers can only be used for relative priority of layers. They convey a layer hierarchy with TID=0 and LID=0 identifying the base layer. Higher values of TID identify higher temporal layers with higher frame rates. Higher values of LID identify higher spatial and/or quality layers with higher resolutions and/or bitrates.

With further information, for example, possible future RTCP SDES items that convey full layer structure information, it may be possible to map these TIDs and LIDs to specific frame rates, resolutions and bitrates. Such additional layer information may be useful for forwarding decisions in the RTP switch, but is beyond the scope of this memo. The relative layer information is still useful for many selective forwarding decisions even without such additional layer information.

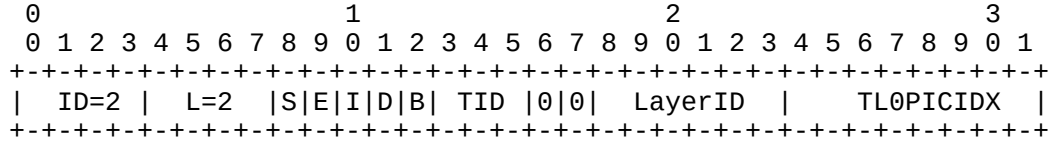
3.2.1. Layer ID Mappings for Scalable Streams

3.2.1.1. H265 LID Mapping

The following shows the H265 [RFC7798] LayerID (6 bits) and TID (3 bits) from the NAL unit header mapped to the generic LID and TID fields.

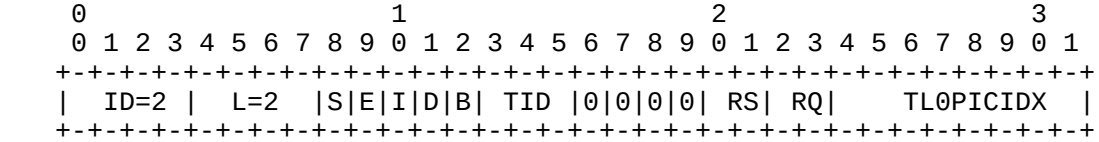
The I bit MUST be 1 when the NAL unit type is 16-23 (inclusive), otherwise it MUST be 0.

The S and E bits MUST match the corresponding bits in PACI:PHES:TSCI payload structures.



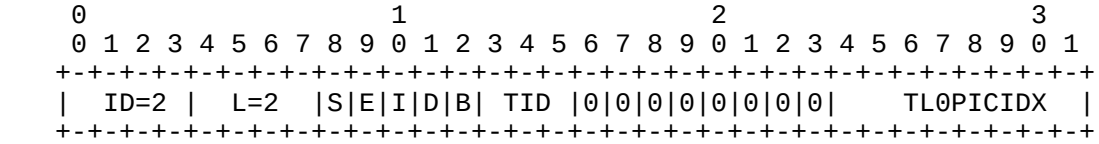
3.2.1.2. VP9 LID Mapping

The following shows VP9 Layer encoding information (4 bits for spatial and quality, 3 bits for temporal layer) mapped to the generic LID and TID fields.



3.2.1.3. VP8 LID Mapping

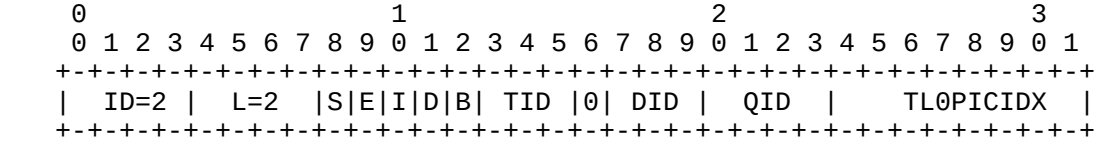
The following shows the header extension for VP8 that contains only temporal layer information.



3.2.1.4. H264-SVC LID Mapping

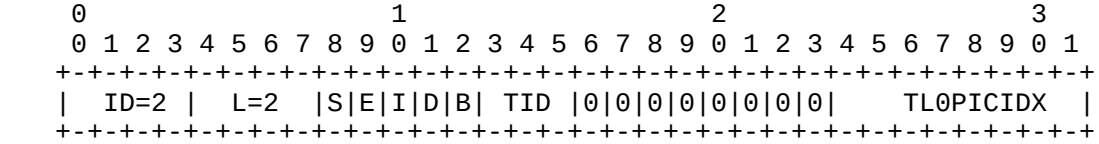
The following shows H264-SVC [RFC6190] Layer encoding information (3 bits for spatial/dependency layer, 4 bits for quality layer and 3 bits for temporal layer) mapped to the generic LID and TID fields.

The S, E, I and D bits MUST match the corresponding bits in PACSI payload structures.



3.2.1.5. H264 (AVC) LID Mapping

The following shows the header extension for H264 (AVC) that contains only temporal layer information.



3.3. Signaling Information

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdrext:framemarking". It does not contain any extension attributes.

An example attribute line in SDP:

```
a=extmap:3 urn:ietf:params:rtp-hdrext:framemarking
```

3.4. Usage Considerations

The header extension values MUST represent what is already in the RTP payload.

When a RTP switch needs to discard a received video frame due to congestion control considerations, it is RECOMMENDED that it preferably drop frames marked with the "discardable" bit.

When a RTP switch wants to forward a new video stream to a receiver, it is RECOMMENDED to select the new video stream from the first switching point (I bit set) and forward the same. A RTP switch can request a media source to generate a switching point for H.264 by sending Full Intra Request (RTCP FIR) as defined in [RFC5104], for example.

3.4.1. Relation to Layer Refresh Request (LRR)

Receivers can use the Layer Refresh Request (LRR) [I-D.ietf-avtext-lrr] RTCP feedback message to upgrade to a higher layer in scalable encodings. The TID/LID values and formats used in LRR messages correspond to the same values and formats specified in Section 3.2.

4. Security Considerations

In the Secure Real-Time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but usually not encrypted. When header extensions are used some of the payload type information are exposed and visible to middle boxes. The encrypted media data is not exposed, so this is not seen as a high risk exposure.

5. Acknowledgements

Many thanks to Bernard Aboba, Jonathan Lennox, and Stephan Wenger for their inputs.

6. IANA Considerations

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

Extension URI: urn:ietf:params:rtp-hdrext:framemarkinginfo
Description: Frame marking information for video streams
Contact: mzanaty@cisco.com
Reference: RFC XXXX

Note to RFC Editor: please replace RFC XXXX with the number of this RFC.

7. References

7.1. 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>.

7.2. Informative References

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

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

[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>.

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

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

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

[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>.

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <http://www.rfc-editor.org/info/rfc6184>.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <http://www.rfc-editor.org/info/rfc6190>.

[RFC7798] Wang, Y., Sanchez, Y., Schierl, T., Wenger, S., and M. Hannuksela, "RTP Payload Format for High Efficiency Video Coding (HEVC)", RFC 7798, DOI 10.17487/RFC7798, March 2016, <http://www.rfc-editor.org/info/rfc7798>.

[I-D.ietf-avtext-lrr] Lennox, J., Hong, D., Uberti, J., Holmer, S., and M. Flodman, "The Layer Refresh Request (LRR) RTCP Feedback Message", draft-ietf-avtext-lrr-03 (work in progress), July 2016.

Authors' Addresses

Espen Berger
Cisco Systems

Phone: +47 98228179
Email: espeberg@cisco.com

Suhas Nandakumar
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: snandaku@cisco.com

Mo Zanaty
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: mzanaty@cisco.com

Payload Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2017

J. Lennox
D. Hong
Vidyo
J. Uberti
S. Holmer
M. Flodman
Google

June 15, 2017

The Layer Refresh Request (LRR) RTCP Feedback Message
draft-ietf-avtext-lrr-06

Abstract

This memo describes the RTCP Payload-Specific Feedback Message "Layer Refresh Request" (LRR), which can be used to request a state refresh of one or more substreams of a layered media stream. It also defines its use with several RTP payloads for scalable media formats.

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 December 17, 2017.

Table of Contents

1. Introduction

This memo describes an RTCP [RFC3550] Payload-Specific Feedback Message [RFC4585] "Layer Refresh Request" (LRR). It is designed to allow a receiver of a layered media stream to request that one or more of its substreams be refreshed, such that it can then be decoded by an endpoint which previously was not receiving those layers, without requiring that the entire stream be refreshed (as it would be if the receiver sent a Full Intra Request (FIR); [RFC5104] see also [RFC8082]).

The feedback message is applicable both to temporally and spatially scaled streams, and to both single-stream and multi-stream scalability modes.

2. Conventions, Definitions and Acronyms

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

2.1. Terminology

A "Layer Refresh Point" is a point in a scalable stream after which a decoder, which previously had been able to decode only some (possibly none) of the available layers of stream, is able to decode a greater number of the layers.

For spatial (or quality) layers, layer refresh typically requires that a spatial layer be encoded in a way that references only lower-layer subpictures of the current picture, not any earlier pictures of that spatial layer. Additionally, the encoder must promise that no earlier pictures of that spatial layer will be used as reference in the future.

In a layer refresh, however, other layers than the ones requested for refresh may still maintain dependency on earlier content of the stream. This is the difference between a layer refresh and a Full Intra Request [RFC5104]. This minimizes the coding overhead of refresh to only those parts of the stream that actually need to be refreshed at any given time.

An illustration of spatial layer refresh of an enhancement layer is shown below.

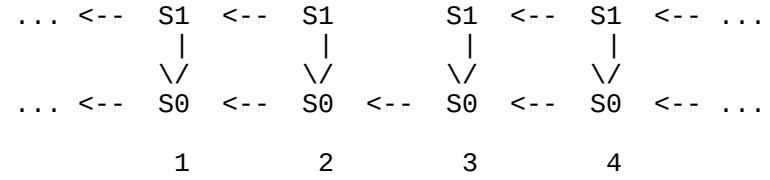


Figure 1

In Figure 1, frame 3 is a layer refresh point for spatial layer S1; a decoder which had previously only been decoding spatial layer S0 would be able to decode layer S1 starting at frame 3.

An illustration of spatial layer refresh of a base layer is shown below.

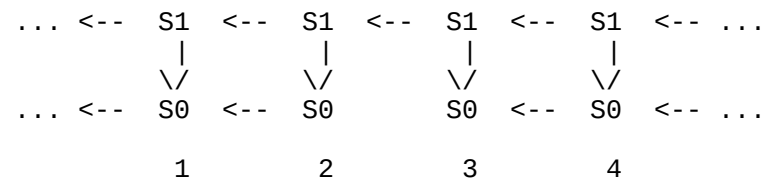


Figure 2

In Figure 2, frame 3 is a layer refresh point for spatial layer S0; a decoder which had previously not been decoding the stream at all could decode layer S0 starting at frame 3.

For temporal layers, layer refresh requires that the layer be "temporally nested", i.e. use as reference only earlier frames of a lower temporal layer, not any earlier frames of this temporal layer, and also promise that no future frames of this temporal layer will reference frames of this temporal layer before the refresh point. In many cases, the temporal structure of the stream will mean that all frames are temporally nested, in which case decoders will have no need to send LRR messages for the stream.

An illustration of temporal layer refresh is shown below.

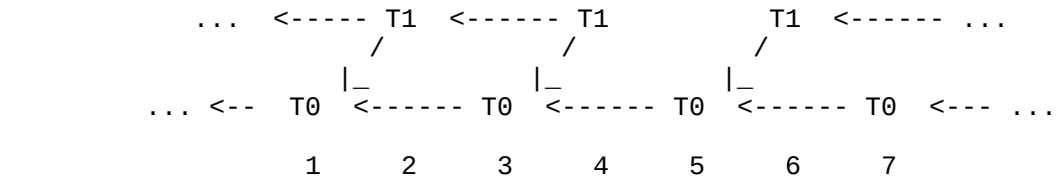


Figure 3

In Figure 3, frame 6 is a layer refresh point for temporal layer T1; a decoder which had previously only been decoding temporal layer T0 would be able to decode layer T1 starting at frame 6.

An illustration of an inherently temporally nested stream is shown below.

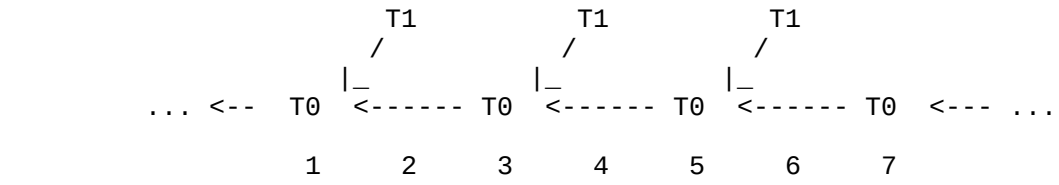


Figure 4

In Figure 4, the stream is temporally nested in its ordinary structure; a decoder receiving layer T0 can begin decoding layer T1 at any point.

A "Layer Index" is a numeric label for a specific spatial and temporal layer of a scalable stream. It consists of the pair of a "temporal ID" identifying the temporal layer, and a "layer ID" identifying the spatial or quality layer. The details of how layers

of a scalable stream are labeled are codec-specific. Details for several codecs are defined in Section 4.

3. Layer Refresh Request

A layer refresh frame can be requested by sending a Layer Refresh Request (LRR), which is an RTP Control Protocol (RTCP) [RFC3550] payload-specific feedback message [RFC4585] asking the encoder to encode a frame which makes it possible to upgrade to a higher layer. The LRR contains one or two tuples, indicating the temporal and spatial layer the decoder wants to upgrade to, and (optionally) the currently highest temporal and spatial layer the decoder can decode.

The specific format of the tuples, and the mechanism by which a receiver recognizes a refresh frame, is codec-dependent. Usage for several codecs is discussed in Section 4.

LRR follows the model of the Full Intra Request (FIR) [RFC5104] (Section 3.5.1) for its retransmission, reliability, and use in multipoint conferences.

The LRR message is identified by RTCP packet type value PT=PSFB and FMT=TBD. The FCI field MUST contain one or more LRR entries. Each entry applies to a different media sender, identified by its SSRC.

[NOTE TO RFC Editor: Please replace "TBD" with the IANA-assigned payload-specific feedback number.]

3.1. Message Format

The Feedback Control Information (FCI) for the Layer Refresh Request consists of one or more FCI entries, the content of which is depicted in Figure 5. The length of the LRR feedback message MUST be set to 2+3*N, where N is the number of FCI entries.

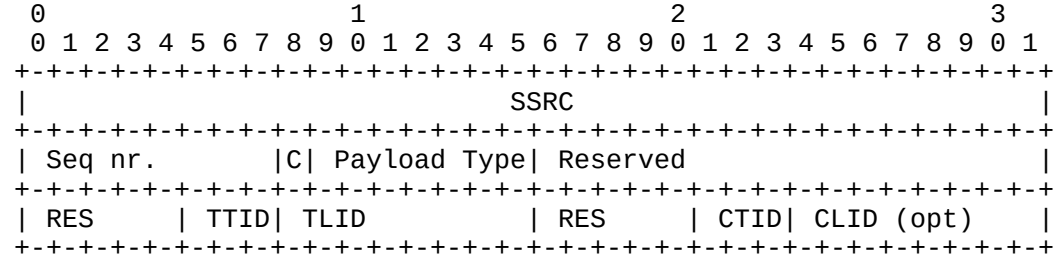


Figure 5

SSRC (32 bits) The SSRC value of the media sender that is requested to send a layer refresh point.

Seq nr. (8 bits) Command sequence number. The sequence number space is unique for each pairing of the SSRC of command source and the SSRC of the command target. The sequence number SHALL be increased by 1 modulo 256 for each new command. A repetition SHALL NOT increase the sequence number. The initial value is arbitrary.

C (1 bit) A flag bit indicating whether the "Current Temporal Layer ID (CTID)" and "Current Layer ID (CLID)" fields are present in the FCI. If this bit is 0, the sender of the LRR message is requesting refresh of all layers up to and including the target layer.

Payload Type (7 bits) The RTP payload type for which the LRR is being requested. This gives the context in which the target layer index is to be interpreted.

Reserved (RES) (16 bits / 5 bits / 5 bits) All bits SHALL be set to 0 by the sender and SHALL be ignored on reception.

Target Temporal Layer ID (TTID) (3 bits) The temporal ID of the target layer for which the receiver wishes a refresh point.

Target Layer ID (TLID) (8 bits) The layer ID of the target spatial or quality layer for which the receiver wishes a refresh point. Its format is dependent on the payload type field.

Current Temporal Layer ID (CTID) (3 bits) If C is 1, the ID of the current temporal layer being decoded by the receiver. This message is not requesting refresh of layers at or below this layer. If C is 0, this field SHALL be set to 0 by the sender and SHALL be ignored on reception.

Current Layer ID (CLID) (8 bits) If C is 1, the layer ID of the current spatial or quality layer being decoded by the receiver. This message is not requesting refresh of layers at or below this layer. If C is 0, this field SHALL be set to 0 by the sender and SHALL be ignored on reception.

When C is 1, TTID MUST NOT be less than CTID, and TLID MUST NOT be less than CLID; at least one of TTID or TLID MUST be greater than CTID or CLID respectively. That is to say, the target layer index <TTID, TLID> MUST be a layer upgrade from the current layer index <CTID, CLID>. A sender MAY request an upgrade in both temporal and spatial/quality layers simultaneously.

Note: the syntax of the TTID, TLID, CTID, and CLID fields match, by design, the TID and LID fields in [I-D.ietf-avtext-framemarking].

3.2. Semantics

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the request, and the "SSRC of media source" is not used and SHALL be set to 0. The SSRCs of the media senders to which the LRR command applies are in the corresponding FCI entries. A LRR message MAY contain requests to multiple media senders, using one FCI entry per target media sender.

Upon reception of LRR, the encoder MUST send a decoder refresh point (see Section 2.1) as soon as possible.

The sender MUST respect bandwidth limits provided by the application of congestion control, as described in Section 5 of [RFC5104]. As layer refresh points will often be larger than non-refreshing frames, this may restrict a sender’s ability to send a layer refresh point quickly.

LRR MUST NOT be sent as a reaction to picture losses -- it is RECOMMENDED to use PLI [RFC4585] instead. LRR SHOULD be used only in situations where not sending a layer refresh point would render the video unusable for the users.

4. Usage with specific codecs

In order for LRR to be used with a scalable codec, the format of the temporal and layer ID fields (for both the target and current layer indices) needs to be specified for that codec’s RTP packetization. New RTP packetization specifications for scalable codecs SHOULD define how this is done. (The VP9 payload [I-D.ietf-payload-vp9], for instance, has done so.) If the payload also specifies how it is used with the Frame Marking RTP Header Extension [I-D.ietf-avtext-framemarking], the syntax MUST be defined in the same manner as the TID and LID fields in that header.

4.1. H264 SVC

H.264 SVC [RFC6190] defines temporal, dependency (spatial), and quality scalability modes.

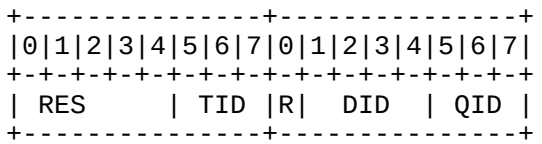


Figure 6

Figure 6 shows the format of the layer index fields for H.264 SVC streams. The "R" and "RES" fields MUST be set to 0 on transmission and ignored on reception. See [RFC6190] Section 1.1.3 for details on the DID, QID, and TID fields.

A dependency or quality layer refresh of a given layer in H.264 SVC can be identified by the "I" bit (idr_flag) in the extended NAL unit header, present in NAL unit types 14 (prefix NAL unit) and 20 (coded scalable slice). Layer refresh of the base layer can also be identified by its NAL unit type of its coded slices, which is "5" rather than "1". A dependency or quality layer refresh is complete once this bit has been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

Note that as the "I" bit in a PACSI header is set if the corresponding bit is set in any of the aggregated NAL units it describes; thus, it is not sufficient to identify layer refresh when NAL units of multiple dependency or quality layers are aggregated.

In H.264 SVC, temporal layer refresh information can be determined from various Supplemental Encoding Information (SEI) messages in the bitstream.

Whether an H.264 SVC stream is scalably nested can be determined from the Scalability Information SEI message’s temporal_id_nesting flag. If this flag is set in a stream’s currently applicable Scalability Information SEI, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point. (The Scalability Information SEI message may also be available in the signaling negotiation of H.264 SVC, as the sprop-scalability-info parameter.)

If a stream’s temporal_id_nesting flag is not set, the Temporal Level Switching Point SEI message identifies temporal layer switching points. A temporal layer refresh is satisfied when this SEI message is present in a frame with the target layer index, if the message’s delta_frame_num refers to a frame with the requested current layer index. (Alternately, temporal layer refresh can also be satisfied by a complete state refresh, such as an IDR.) Senders which support

receiving LRR for non-temporally-nested streams MUST insert Temporal Level Switching Point SEI messages as appropriate.

4.2. VP8

The VP8 RTP payload format [RFC7741] defines temporal scalability modes. It does not support spatial scalability.

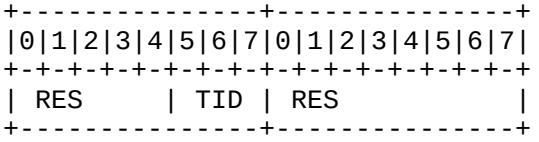


Figure 7

Figure 7 shows the format of the layer index field for VP8 streams. The "RES" fields MUST be set to 0 on transmission and be ignored on reception. See [RFC7741] Section 4.2 for details on the TID field.

A VP8 layer refresh point can be identified by the presence of the "Y" bit in the VP8 payload header. When this bit is set, this and all subsequent frames depend only on the current base temporal layer. On receipt of an LRR for a VP8 stream, A sender which supports LRR MUST encode the stream so it can set the Y bit in a packet whose temporal layer is at or below the target layer index.

Note that in VP8, not every layer switch point can be identified by the Y bit, since the Y bit implies layer switch of all layers, not just the layer in which it is sent. Thus the use of LRR with VP8 can result in some inefficiency in transmision. However, this is not expected to be a major issue for temporal structures in normal use.

4.3. H265

The initial version of the H.265 payload format [RFC7798] defines temporal scalability, with protocol elements reserved for spatial or other scalability modes (which are expected to be defined in a future version of the specification).

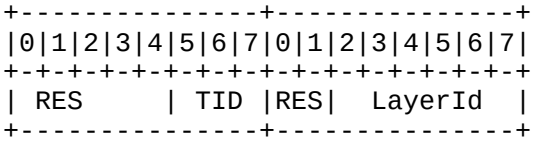


Figure 8

Figure 8 shows the format of the layer index field for H.265 streams. The "RES" fields MUST be set to 0 on transmission and ignored on reception. See [RFC7798] Section 1.1.4 for details on the LayerId and TID fields.

H.265 streams signal whether they are temporally nested, using the vps_temporal_id_nesting_flag in the Video Parameter Set (VPS), and the sps_temporal_id_nesting_flag in the Sequence Parameter Set (SPS). If this flag is set in a stream's currently applicable VPS or SPS, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point.

If a stream's sps_temporal_id_nesting_flag is not set, the NAL unit types 2 to 5 inclusively identify temporal layer switching points. A layer refresh to any higher target temporal layer is satisfied when a NAL unit type of 4 or 5 with TID equal to 1 more than current TID is seen. Alternatively, layer refresh to a target temporal layer can be incrementally satisfied with NAL unit type of 2 or 3. In this case, given current TID = TO and target TID = TN, layer refresh to TN is satisfied when NAL unit type of 2 or 3 is seen for TID = T1, then TID = T2, all the way up to TID = TN. During this incremental process, layer refresh to TN can be completely satisfied as soon as a NAL unit type of 2 or 3 is seen.

Of course, temporal layer refresh can also be satisfied whenever any Intra Random Access Point (IRAP) NAL unit type (with values 16-23, inclusively) is seen. An IRAP picture is similar to an IDR picture in H.264 (NAL unit type of 5 in H.264) where decoding of the picture can start without any older pictures.

In the (future) H.265 payloads that support spatial scalability, a spatial layer refresh of a specific layer can be identified by NAL units with the requested layer ID and NAL unit types between 16 and 21 inclusive. A dependency or quality layer refresh is complete once NAL units of this type have been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

5. Usage with different scalability transmission mechanisms

Several different mechanisms are defined for how scalable streams can be transmitted in RTP. The RTP Taxonomy [RFC7656] Section 3.7 defines three mechanisms: Single RTP Stream on a Single Media Transport (SRST), Multiple RTP Streams on a Single Media Transport (MRST), and Multiple RTP Streams on Multiple Media Transports (MRMT).

The LRR message is applicable to all these mechanisms. For MRST and MRMT mechanisms, the "media source" field of the LRR FCI is set to

the SSRC of the RTP stream containing the layer indicated by the Current Layer Index (if "C" is 1), or the stream containing the base encoded stream (if "C" is 0). For MRMT, it is sent on the RTP session on which this stream is sent. On receipt, the sender MUST refresh all the layers requested in the stream, simultaneously in decode order.

6. Security Considerations

All the security considerations of FIR feedback packets [RFC5104] apply to LRR feedback packets as well. Additionally, media senders receiving LRR feedback packets MUST validate that the payload types and layer indices they are receiving are valid for the stream they are currently sending, and discard the requests if not.

7. SDP Definitions

Section 7 of [RFC5104] defines SDP procedures for indicating and negotiating support for codec control messages (CCM) in SDP. This document extends this with a new codec control command, "lrr", which indicates support of the Layer Refresh Request (LRR).

Figure 9 gives a formal Augmented Backus-Naur Form (ABNF) [RFC5234] showing this grammar extension, extending the grammar defined in [RFC5104].

```
rtcp-fb-ccm-param =/ SP "lrr"    ; Layer Refresh Request
```

Figure 9: Syntax of the "lrr" ccm

The Offer-Answer considerations defined in [RFC5104] Section 7.2 apply.

8. IANA Considerations

This document defines a new entry to the "Codec Control Messages" subregistry of the "Session Description Protocol (SDP) Parameters" registry, according to the following data:

Value name: lrr

Long name: Layer Refresh Request Command

Usable with: ccm

Reference: RFC XXXX

This document also defines a new entry to the "FMT Values for PSFB Payload Types" subregistry of the "Real-Time Transport Protocol (RTP) Parameters" registry, according to the following data:

Name: LRR

Long Name: Layer Refresh Request Command

Value: TBD

Reference: RFC XXXX

9. References

9.1. Normative References

[I-D.ietf-avtext-framemarking]
Berger, E., Nandakumar, S., and M. Zanaty, "Frame Marking RTP Header Extension", draft-ietf-avtext-framemarking-04 (work in progress), March 2017.

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

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

[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>.

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

[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>.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <http://www.rfc-editor.org/info/rfc6190>.

[RFC7741] Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", RFC 7741, DOI 10.17487/RFC7741, March 2016, <http://www.rfc-editor.org/info/rfc7741>.

[RFC7798] Wang, Y., Sanchez, Y., Schierl, T., Wenger, S., and M. Hannuksela, "RTP Payload Format for High Efficiency Video Coding (HEVC)", RFC 7798, DOI 10.17487/RFC7798, March 2016, <http://www.rfc-editor.org/info/rfc7798>.

9.2. Informative References

[I-D.ietf-payload-vp9] Uberti, J., Holmer, S., Flodman, M., Lennox, J., and D. Hong, "RTP Payload Format for VP9 Video", draft-ietf-payload-vp9-03 (work in progress), March 2017.

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

[RFC8082] Wenger, S., Lennox, J., Burman, B., and M. Westerlund, "Using Codec Control Messages in the RTP Audio-Visual Profile with Feedback with Layered Codecs", RFC 8082, DOI 10.17487/RFC8082, March 2017, <http://www.rfc-editor.org/info/rfc8082>.

Authors' Addresses

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

Email: jonathan@vidyo.com

Danny Hong
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: danny@vidyo.com

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name

Stefan Holmer
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: holmer@google.com

Magnus Flodman
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: mflodman@google.com

AVTCore Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 28, 2017

J. Lennox
Vidyo

March 27, 2017

DTLS/SRTP Protection Profiles for 256-bit AES-CTR Encryption
draft-lennox-avtcore-dtls-srtp-bigaes-00

Abstract

This memo defines Datagram Transport Layer Security (DTLS) Secure Real-time Transport Protocol (SRTP) Protection Profiles for 256-bit Advanced Encryption Standard (AES) Counter Mode.

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 28, 2017.

Copyright Notice

Table of Contents

1. Introduction

This memo defines Datagram Transport Layer Security (DTLS) Secure Real-time Transport Protocol (SRTP) Protection Profiles for 256-bit Advanced Encryption Standard (AES) Counter Mode.

DTLS-based key establishment for SRTP is defined in [RFC5764]. The use of AES-256 counter mode with SRTP is defined in [RFC6188].

The draft document that became [RFC5764] initially defined protection profiles for AES-256; they were removed because the document that became [RFC6188] was not yet ready. However, the definitions of the protection profiles were not transfered to the [RFC6188] drafts, apparently as an oversight. This document restores those codepoints, with their original values.

1.1. Motivation

The question might arise as to why this is necessary. [RFC7714] defines the use of AES-256 with Galois Counter Mode, and current thought is that Galois Counter Mode is preferable to Counter Mode plus HMAC-based authentication.

The reason is to minimize the difficulty of moving implementations away from Security Descriptions-based keying [RFC4568]. Use of Security Descriptions is strongly discouraged, as its security properties are much weaker than those of DTLS/SRTP. However, as [RFC6188] defines Security Descriptions signaling elements for AES-256-CTR, existing implementations use them to negotiate the use of these crypto suites, and many of the these implementations do not have Galois Counter Mode cryptography implemented (or certified). Thus, defining AES-256-CTR codepoints for DTLS/SRTP allows these implementations to continue using their existing SRTP cryptography while moving to a more secure keying protocol.

2. Conventions, Definitions and Acronyms

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. SRTP Protection Profiles

A DTLS-SRTP SRTP Protection Profile defines the parameters and options that are in effect for the SRTP processing. This document defines the following SRTP protection profiles.

```
SRTPProtectionProfile SRTP_AES256_CM_SHA1_80 = {0x00, 0x03};
SRTPProtectionProfile SRTP_AES256_CM_SHA1_32 = {0x00, 0x04};
```

The following list indicates the SRTP transform parameters for each protection profile. The parameters cipher_key_length, cipher_salt_length, auth_key_length, and auth_tag_length express the number of bits in the values to which they refer. The maximum_lifetime parameter indicates the maximum number of packets that can be protected with each single set of keys when the parameter profile is in use. All of these parameters apply to both RTP and RTCP, unless the RTCP parameters are separately specified.

All of the crypto algorithms in these profiles are from [RFC6188].

```
SRTP_AES256_CM_HMAC_SHA1_80
      cipher:  AES_256_CM
      cipher_key_length:  256
      cipher_salt_length:  112
      maximum_lifetime:  2^31
      auth_function:  HMAC-SHA1
      auth_key_length:  160
      auth_tag_length:  80
SRTP_AES256_CM_HMAC_SHA1_32
      cipher:  AES_256_CM
      cipher_key_length:  256
      cipher_salt_length:  112
      maximum_lifetime:  2^31
      auth_function:  HMAC-SHA1
      auth_key_length:  160
      auth_tag_length:  32
      RTCP auth_tag_length:  80
```

With both of these SRTP Parameter profiles, the following SRTP options are in effect:

- The TLS Key Derivation Function (KDF) is used to generate keys to feed into the SRTP KDF.
- The Key Derivation Rate (KDR) is equal to zero. Thus, keys are not re-derived based on the SRTP sequence number.
- The key derivation procedures from Section 3 of AES_256_CM_KDF [RFC6188] are used.
- For all other parameters, (in particular, SRTP replay window size and FEC order) the default values are used.

If values other than the defaults for these parameters are required, they can be enabled by writing a separate specification specifying SDP syntax to signal them.

4. Security Considerations

This document defines security mechanisms. No additional security issues beyond those of [RFC5764] and [RFC6188] apply.

5. IANA Considerations

IANA is requested to add the SRTP Protection Profiles defined in Section 3 to the DTLS SRTPProtectionProfile registry.

6. References

6.1. 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>.

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

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

[RFC6188] McGrew, D., "The Use of AES-192 and AES-256 in Secure RTP", RFC 6188, DOI 10.17487/RFC6188, March 2011, <http://www.rfc-editor.org/info/rfc6188>.

6.2. Informative References

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <http://www.rfc-editor.org/info/rfc4568>.

[RFC7714] McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)", RFC 7714, DOI 10.17487/RFC7714, December 2015, <http://www.rfc-editor.org/info/rfc7714>.

Author’s Address

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

Email: jonathan@vidyo.com