

AVTCORE	J. Lennox
Internet-Draft	Vidyo
Intended status: Standards Track	October 29, 2011
Expires: May 01, 2012	

Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)

draft-ietf-avtcore-srtp-encrypted-header-ext-01

[Abstract](#)

The Secure Real-Time Transport Protocol (SRTP) provides authentication, but not encryption, of the headers of Real-Time Transport Protocol (RTP) packets. However, RTP header extensions may carry sensitive information for which participants in multimedia sessions want confidentiality. This document provides a mechanism, extending the mechanisms of SRTP, to selectively encrypt RTP header extensions in SRTP.

[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 01, 2012.

[Copyright Notice](#)

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

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

[Table of Contents](#)

- *1. [Introduction](#)

- *2. [Terminology](#)
- *3. [Encryption Mechanism](#)
- *3.1. [Example Encryption Mask](#)
- *4. [Signaling \(Setup\) Information](#)
- *4.1. [Backward compatibility](#)
- *5. [Security Considerations](#)
- *6. [IANA Considerations](#)
- *7. [Acknowledgments](#)
- *8. [References](#)
- *8.1. [Normative References](#)
- *8.2. [Informative References](#)
- *Appendix A. [Test Vectors](#)
- *Appendix A.1. [Key derivation test vectors](#)
- *Appendix A.2. [Header Encryption Test Vectors using AES-CM](#)
- *Appendix B. [Changes From Earlier Versions](#)
- *Appendix B.1. [Changes from draft-ietf-avtcore -00](#)
- *Appendix B.2. [Changes from draft-lennox-avtcore -00](#)
- *Appendix B.3. [Changes from draft-lennox-avt -02](#)
- *Appendix B.4. [Changes From Individual Submission Draft -01](#)
- *Appendix B.5. [Changes From Individual Submission Draft -00](#)
- *[Author's Address](#)

[1. Introduction](#)

The [Secure Real-Time Transport Protocol](#) [RFC3711] specification provides confidentiality, message authentication, and replay protection for multimedia payloads sent using of the [Real-Time Protocol \(RTP\)](#) [RFC3550]. However, in order to preserve RTP header compression efficiency, SRTP provides only authentication and replay protection for the headers of RTP packets, not confidentiality.

For the standard portions of an RTP header, this does not normally present a problem, as the information carried in an RTP header does not provide much information beyond that which an attacker could infer by observing the size and timing of RTP packets. Thus, there is little need for confidentiality of the header information.

However, this is not necessarily true for information carried in RTP header extensions. A number of recent proposals for header extensions using the [General Mechanism for RTP Header Extensions](#) [RFC5285] carry information for which confidentiality could be desired or essential. Notably, two recent drafts ([\[I-D.ietf-avtext-client-to-mixer-audio-level\]](#) and [\[I-D.ietf-avtext-mixer-to-client-audio-level\]](#)) carry information about per-packet sound levels of the media data carried in the RTP payload, and exposing this to an eavesdropper may be unacceptable in many circumstances.

This document, therefore, defines a mechanism by which encryption can be applied to RTP header extensions when they are transported using SRTP. As an RTP sender may wish some extension information to be sent in the clear (for example, it may be useful for a network monitoring device to be aware of [RTP transmission time offsets](#) [RFC5450]), this mechanism can be selectively applied to a subset of the header extension elements carried in an SRTP packet.

[2. Terminology](#)

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [RFC2119] and indicate requirement levels for compliant implementations.

[3. Encryption Mechanism](#)

Encrypted header extension elements are carried in the same manner as non-encrypted header extension elements, as defined by [\[RFC5285\]](#). The (one- or two-byte) header of the extension elements is not encrypted, nor is any of the header extension padding. If multiple different header extension elements are being encrypted, they have separate element identifier values, just as they would if they were not encrypted; similarly, encrypted and non-encrypted header extension elements have separate identifier values.

Encrypted extension headers are only carried in packets encrypted using the [Secure Real-Time Transport Protocol](#) [RFC3711]. To encrypt (or decrypt) encrypted extension headers, an SRTP participant first uses the SRTP Key Derivation Algorithm, specified in Section 4.3.1 of [\[RFC3711\]](#), to generate header encryption and header salting keys, using the same pseudo-random function family as are used for the key derivation for the SRTP session. These keys are derived as follows:

*k_he (SRTP header encryption): <label> = 0x06, n=n_e.

*k_hs (SRTP header salting key): <label> = 0x07, n=n_s.

where n_e and n_s are from the cryptographic context: the same size encryption key and salting key are used as are used for the SRTP payload. (Note that since RTP headers, including extension headers, are authenticated in SRTP, no new authentication key is needed for extension headers.)

For SRTP encryption transforms that operate by generating a keystream, a header keystream is generated for each packet containing an encrypted header, using the same encryption transform and Initialization Vector (IV) as is used for the SRTP payload, except that the SRTP encryption and salting keys k_e and k_s are replaced by the SRTP header encryption and header salting keys k_he and k_hs, respectively.

The AES-CM and AES-f8 transforms defined in [\[RFC3711\]](#) both operate in this keystream mode, as do the AES_192_CM and AES_256_CM transforms defined in [\[RFC6188\]](#). For other transforms (for example, Authenticated Encryption with Associated Data (AEAD) cryptographic transforms, such as [AES_GCM and AES_CCM](#) [*I-D.ietf-avt-srtp-aes-gcm*]) their usage of header extensions MUST be specified explicitly. (As of this writing, it is believed that it will be sufficient for SRTP packets protected with AEAD transforms to use a CM transform with equivalent algorithms and key lengths for their encrypted headers; however, this guidance is not normative.)

Once the header keystream is generated, the SRTP participant then computes an encryption mask for the header extension, identifying the portions of the header extension that are, or are to be, encrypted. This encryption mask corresponds to the entire payload of each header extension element that is encrypted. It does not include any non-encrypted header extension elements, any extension element headers, or any padding octets. The encryption mask has all-bits-1 octets (i.e., hexadecimal 0xff) for header extension octets which are to be encrypted, and all-bits-0 octets for header extension octets which are not to be.

For those octets indicated in the encryption mask, the SRTP participant bitwise exclusive-ors the header extension with the keystream to produce the ciphertext version of the header extension. Those octets not indicated in the encryption mask are left unmodified. Thus, conceptually, the encryption mask is logically ANDed with the keystream to produce a masked keystream. The sender and receiver MUST use the same encryption mask. The set of extension elements to be encrypted is communicated between the sender and the receiver using the signaling mechanisms described in [Section 4](#).

The SRTP authentication tag is computed across the encrypted header extension, i.e., the data that is actually transmitted on the wire. Thus, header extension encryption MUST be done before the authentication tag is computed, and authentication tag validation MUST be done on the encrypted header extensions. For receivers, header extension decryption SHOULD be done only after the receiver has validated the packet's message authentication tag, and the receiver

MUST NOT take any actions based on decrypted headers that could affect the security or proper functioning of the system, prior to validating the authentication tag.

3.1. Example Encryption Mask

If a sender wished to send a header extension containing an encrypted [SMPTE timecode](#) [RFC5484] with ID 1, a plaintext [transmission time offset](#) [RFC5450] with ID 2, an encrypted [audio level indication](#) [I-D.ietf-avtext-client-to-mixer-audio-level] with ID 3, and an encrypted [NTP Timestamp](#) [RFC6051] with ID 4, the plaintext RTP header extension might look like this:

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| ID=1 | len=7 |      SMPTE timecode (long form)      |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|      SMPTE timecode (continued)      |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| SMPTE (cont'd)| ID=2 | len=2 | toffset              |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| toffset (ct'd)| ID=3 | len=0 | audio level      | ID=4 | len=6 |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|      NTP Timestamp (Variant B)      |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|      NTP Timestamp (Variant B, cont.)      | padding = 0 |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

The corresponding encryption mask would then be:

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|0 0 0 0 0 0 0 0|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|1 1 1 1 1 1 1 1|0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0|1 1 1 1 1 1 1 1|0 0 0 0 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|0 0 0 0 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

In the mask, the octets corresponding to the payloads of the encrypted header extension elements are set to all-1 values, and octets corresponding to non-encrypted elements, element headers, and header extension padding are set to all-0 values.

4. Signaling (Setup) Information

Encrypted header extension elements are signaled in the SDP extmap attribute, using the URI "urn:ietf:params:rtp-hdext:encrypt", followed by the URI of the header extension element being encrypted as well as any extensionattributes that extension normally takes. Thus, for example, to signal an SRTP session using encrypted [SMPTE timecodes \[RFC5484\]](#), while simultaneously signaling plaintext [transmission time offsets \[RFC5450\]](#), an SDP document could contain (line breaks added for formatting):

```
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32 \
    inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=extmap:1 urn:ietf:params:rtp-hdext:encrypt \
    urn:ietf:params:rtp-hdext:smppte-tc 25@600/24
a=extmap:2 urn:ietf:params:rtp-hdext:toffset
```

This example uses [SDP Security Descriptions \[RFC4568\]](#) for SRTP keying, but this is merely for illustration; any SRTP keying mechanism to establish session keys will work.

The extmap SDP attribute is defined in [\[RFC5285\]](#) as being either a session or media attribute. If the extmap for an encrypted header extension is specified as a media attribute, it MUST only be specified for media which use SRTP-based RTP profiles. If such an extmap is specified as a session attribute, there MUST be at least one media in the SDP session which uses an SRTP-based RTP profile; the session-level extmap applies to all the SRTP-based media in the session, and MUST be ignored for all other (non-SRTP or non-RTP) media.

4.1. Backward compatibility

Following the procedures in [\[RFC5285\]](#), an SDP endpoint which does not understand the "urn:ietf:params:rtp-hdext:encrypt" extension URI will ignore the extension, and (for SDP offer/answer) negotiate not to use it.

In a negotiated session (whether using offer/answer or some other means), best-effort encryption of a header extension element is possible: an endpoint MAY offer the same header extension element both encrypted and unencrypted. Receivers which understand header extension encryption SHOULD choose the encrypted form and mark the unencrypted form "inactive", unless they have an explicit reason to prefer the unencrypted form. (Note that, as always, users of best-effort

encryption MUST be cautious of bid-down attacks, and ensure, for example, that signaling is integrity-protected.)

5. Security Considerations

The security properties of header extension elements protected by the mechanism in this document are equivalent to those for SRTP payloads. The mechanism defined in this document does not provide confidentiality about which header extension elements are used for a given SRTP packet, only for the content of those header extension elements. This appears to be in the spirit of SRTP itself, which does not encrypt RTP headers. If this is a concern, an alternate mechanism would be needed to provide confidentiality.

For the two-byte-header form of header extension elements (0x100x), this mechanism does not provide any protection to zero-length header extension elements (for which their presence or absence is the only information they carry). It also does not provide any protection for the two-byte-headers' app bits (field 256, the lowest four bits of the "defined by profile" field). Neither of these features are used in for one-byte-header form of header extension elements (0xBEDE), so these limitations do not apply in that case.

This document does not specify the circumstances in which extension header encryption should be used. Documents defining specific header extension elements should provide guidance on when encryption is appropriate for these elements.

If a middlebox does not have access to the SRTP authentication keys, it has no way to verify the authenticity of unencrypted RTP header extension elements (or the unencrypted RTP header), even though it can monitor them. Therefore, such middleboxes MUST treat such headers as untrusted and potentially generated by an attacker.

6. IANA Considerations

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

Extension URI: urn:ietf:params:rtp-hdext:encrypt

Description: Encrypted extension header element

Contact: jonathan@vidyo.com

Reference: RFC XXXX

(Note to the RFC-Editor: please replace "XXXX" with the number of this document prior to publication as an RFC.)

[7. Acknowledgments](#)

Thanks to Roni Even, Kevin Igoe, David McGrew, David Singer, Qin Wu, and Felix Wyss for their comments and suggestions in the development of this specification.

[8. References](#)

[8.1. Normative References](#)

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels" , BCP 14, RFC 2119, March 1997.
[RFC5285]	Singer, D. and H. Desineni, " A General Mechanism for RTP Header Extensions ", RFC 5285, July 2008.
[RFC3711]	Baughner, M., McGrew, D., Naslund, M., Carrara, E. and K. Norrman, " The Secure Real-time Transport Protocol (SRTP) ", RFC 3711, March 2004.
[RFC3550]	Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, " RTP: A Transport Protocol for Real-Time Applications ", STD 64, RFC 3550, July 2003.
[RFC6188]	McGrew, D., " The Use of AES-192 and AES-256 in Secure RTP ", RFC 6188, March 2011.

[8.2. Informative References](#)

[RFC5450]	Singer, D. and H. Desineni, " Transmission Time Offsets in RTP Streams ", RFC 5450, March 2009.
[I-D.ietf-avtext-client-to-mixer-audio-level]	Lennox, J, Ivov, E and E Marocco, " A Real-Time Transport Protocol (RTP) Header Extension for Client-to- Mixer Audio Level Indication ", Internet-Draft draft-ietf-avtext-client-to-mixer-audio-level-06, November 2011.
[RFC4568]	Andreasen, F., Baughner, M. and D. Wing, " Session Description Protocol (SDP) Security Descriptions for Media Streams ", RFC 4568, July 2006.
[I-D.ietf-avtext-mixer-to-client-audio-level]	Ivov, E, Marocco, E and J Lennox, " A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to- Client Audio Level Indication ", Internet-Draft draft-ietf-avtext-mixer-to-client-audio-level-06, November 2011.
[RFC5484]	Singer, D., " Associating Time-Codes with RTP Streams ", RFC 5484, March 2009.
[RFC6051]	Perkins, C. and T. Schierl, " Rapid Synchronisation of RTP Flows ", RFC 6051, November 2010.
[I-D.ietf-avt-srtp-aes-gcm]	McGrew, D and K Igoe, " AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP) ", Internet-Draft draft-ietf-avt-srtp-aes-gcm-02, October 2011.

[Appendix A. Test Vectors](#)

[Appendix A.1. Key derivation test vectors](#)

This section provides test data for the header extension key derivation function, using AES-128 in Counter Mode. (The algorithms and keys used are the same as those for the test vectors in Appendix B.3 of [\[RFC3711\]](#).)

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

Following [\[RFC3711\]](#), the input block for AES-CM is generated by exclusive-oring the master salt with the concatenation of the encryption key label 0x06 with (index DIV kdr), then padding on the right with two null octets (which implements the multiply-by-2¹⁶ operation, see Section 4.3.3 of [\[RFC3711\]](#)). The resulting value is then AES-CM- encrypted using the master key to get the cipher key.

```
index DIV kdr:          00000000000000
label:                06
master salt:          0EC675AD498AFEEBB6960B3AABE6
-----
xor:                  0EC675AD498AFEEDB6960B3AABE6      (x, PRF input)

x*2^16:               0EC675AD498AFEEDB6960B3AABE60000 (AES-CM input)

hdr. cipher key:      549752054D6FB708622C4A2E596A1B93 (AES-CM output)
```

Next, we show how the cipher salt is generated. The input block for AES-CM is generated by exclusive-oring the master salt with the concatenation of the encryption salt label. That value is padded and encrypted as above.

```
index DIV kdr:          00000000000000
label:                07
master salt:          0EC675AD498AFEEBB6960B3AABE6
-----
xor:                  0EC675AD498AFE ECB6960B3AABE6      (x, PRF input)

x*2^16:               0EC675AD498AFE ECB6960B3AABE60000 (AES-CM input)

                        AB01818174C40D39A3781F7C2D270733 (AES-CM output)

hdr. cipher salt:      AB01818174C40D39A3781F7C2D27
```

[Appendix A.2.](#) Header Encryption Test Vectors using AES-CM

This section provides test vectors for the encryption of a header extension, using the AES_CM cryptographic transform. The header extension element is encrypted using the header cipher key and header cipher salt computed in [Appendix Appendix A.1](#).

Session Key: 549752054D6FB708622C4A2E596A1B93
Session Salt: AB01818174C40D39A3781F7C2D27

SSRC: CAFEBABE
Rollover Counter: 00000000
Sequence Number: 1234

Init. Counter: AB018181BE3AB787A3781F7C3F130000

The RTP session was negotiated to indicate that header extension ID values 1, 3 and 4 are encrypted. In hexadecimal, the header extension being encrypted is (spaces added to show the internal structure of the header extension):

17 414273A475262748 22 0000C8 30 8E 46 55996386B395FB 00

This header extension is 24 bytes long. (Its values are intended to represent plausible values of the header extension elements shown in [Section 3.1](#), but their specific meaning is not important for the example.)

In hexadecimal, the corresponding encryption mask selecting the bodies of header extensions 1, 2, and 4 (corresponding to the mask in [Figure 2](#) is:

00 FFFFFFFFFFFFFFFFFF 00 000000 00 FF 00 FFFFFFFFFFFFFFFFFF 00

Finally, we compute the keystream from the session key and the initial counter, apply the mask to the keystream, and then xor the keystream with the plaintext:

Initial keystream: 1E19C8E1D481C779549ED1617AAA1B7A
FC0D933AE7ED6CC8
Mask (Hex): 00FFFFFFFFFFFFFFFFF000000000FF00
FFFFFFFFFFFFFFFFF00
Masked keystream: 0019C8E1D481C7795400000000001B00
FC0D933AE7ED6C00
Plaintext: 17414273A475262748220000C8308E46
55996386B395FB00
Ciphertext: 17588A9270F4E15E1C220000C8309546
A994F0BC54789700

[Appendix B.](#) **Changes From Earlier Versions**

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

[Appendix B.1.](#) **Changes from draft-ietf-avtcore -00**

Clarified usage of Key Derivation Algorithm

Provided non-normative guidance for how to use this mechanism with Authenticated Encryption with Associated Data (AEAD) transforms.

Corrected SMPTE Timecode header extension element in example header extension (it's eight bytes, not sixteen). Added an NTP timestamp to the example to fill it back out to original size.

Specified applicability of the extmap attribute if it's specified as a session-level attribute.

Added description of backward compatibility, including a description of how you can negotiate best-effort encryption.

Added a note to the security considerations about the dangers for middleboxes observing unencrypted headers (both header extension elements and RTP headers) without being able to verify the authentication keys.

Added test vectors.

Added acknowledgments section.

[Appendix B.2.](#) **Changes from draft-lennox-avtcore -00**

*Published as working group item.

*Added discussion of limitations when used with the two-byte-header form of header extension elements.

*Added open issue about how to use this mechanism with Authenticated Encryption with Associated Data (AEAD) transforms.

*Updated references.

[Appendix B.3.](#) **Changes from draft-lennox-avt -02**

*Retargeted at AVTCORE working group.

*Updated references.

[Appendix B.4.](#) **Changes From Individual Submission Draft -01**

*Minor editorial changes.

[Appendix B.5.](#) **Changes From Individual Submission Draft -00**

*Clarified description of encryption mask creation.

*Added example encryption mask.

*Editorial changes.

Author's Address

Jonathan Lennox Lennox Vidyo, Inc. 433 Hackensack Avenue Seventh
Floor Hackensack, NJ 07601 US EMail: jonathan@vidyo.com