

Workgroup: AVTCORE

Internet-Draft: draft-ietf-avtcore-cryptex-08

Updates: [3711](#) (if approved)

Published: 4 August 2022

Intended Status: Standards Track

Expires: 5 February 2023

Authors: J. Uberti    C. Jennings    S. Garcia Murillo  
          Clubhouse    Cisco                Millicast

## **Completely Encrypting RTP Header Extensions and Contributing Sources**

### **Abstract**

While the Secure Real-time Transport Protocol (SRTP) provides confidentiality for the contents of a media packet, a significant amount of metadata is left unprotected, including RTP header extensions and contributing sources (CSRCs). However, this data can be moderately sensitive in many applications. While there have been previous attempts to protect this data, they have had limited deployment, due to complexity as well as technical limitations.

This document updates RFC 3711, the SRTP specification, and defines Cryptex as a new mechanism that completely encrypts header extensions and CSRCs and uses simpler Session Description Protocol (SDP) signaling with the goal of facilitating deployment.

### **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 <https://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 5 February 2023.

### **Copyright Notice**

Copyright (c) 2022 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

(<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
    - [1.1. Problem Statement](#)
    - [1.2. Previous Solutions](#)
    - [1.3. Goals](#)
  - [2. Terminology](#)
  - [3. Design](#)
  - [4. SDP Considerations](#)
  - [5. RTP Header Processing](#)
    - [5.1. Sending](#)
    - [5.2. Receiving](#)
  - [6. Encryption and Decryption](#)
    - [6.1. Packet Structure](#)
    - [6.2. Encryption Procedure](#)
    - [6.3. Decryption Procedure](#)
  - [7. Backwards Compatibility](#)
  - [8. Security Considerations](#)
  - [9. IANA Considerations](#)
    - [9.1. SDP cryptex Attribute](#)
  - [10. Acknowledgements](#)
  - [11. References](#)
    - [11.1. Normative References](#)
    - [11.2. Informative References](#)
- [Appendix A. Test Vectors](#)
- [A.1. AES-CTR](#)
    - [A.1.1. RTP Packet with 1-byte header extension](#)
    - [A.1.2. RTP Packet with 2-byte header extension](#)
    - [A.1.3. RTP Packet with 1-byte header extension and CSRC fields](#)
    - [A.1.4. RTP Packet with 2-byte header extension and CSRC fields](#)
    - [A.1.5. RTP Packet with empty 1-byte header extension and CSRC fields](#)
    - [A.1.6. RTP Packet with empty 2-byte header extension and CSRC fields](#)
  - [A.2. AES-GCM](#)
    - [A.2.1. RTP Packet with 1-byte header extension](#)
    - [A.2.2. RTP Packet with 2-byte header extension](#)
    - [A.2.3. RTP Packet with 1-byte header extension and CSRC fields](#)
    - [A.2.4. RTP Packet with 2-byte header extension and CSRC fields](#)
    - [A.2.5. RTP Packet with empty 1-byte header extension and CSRC fields](#)

## 1. Introduction

### 1.1. Problem Statement

The Secure Real-time Transport Protocol (SRTP) [[RFC3711](#)] mechanism provides message authentication for the entire RTP packet, but only encrypts the RTP payload. This has not historically been a problem, as much of the information carried in the header has minimal sensitivity (e.g., RTP timestamp); in addition, certain fields need to remain as cleartext because they are used for key scheduling (e.g., RTP SSRC and sequence number).

However, as noted in [[RFC6904](#)], the security requirements can be different for information carried in RTP header extensions, including the per-packet sound levels defined in [[RFC6464](#)] and [[RFC6465](#)], which are specifically noted as being sensitive in the Security Considerations section of those RFCs.

In addition to the contents of the header extensions, there are now enough header extensions in active use that the header extension identifiers themselves can provide meaningful information in terms of determining the identity of the endpoint and/or application. Accordingly, these identifiers can be considered a fingerprinting issue.

Finally, the CSRCs included in RTP packets can also be sensitive, potentially allowing a network eavesdropper to determine who was speaking and when during an otherwise secure conference call.

### 1.2. Previous Solutions

Encryption of Header Extensions in SRTP [[RFC6904](#)] was proposed in 2013 as a solution to the problem of unprotected header extension values. However, it has not seen significant adoption, and has a few technical shortcomings.

First, the mechanism is complicated. Since it allows encryption to be negotiated on a per-extension basis, a fair amount of signaling logic is required. And in the SRTP layer, a somewhat complex transform is required to allow only the selected header extension values to be encrypted. One of the most popular SRTP implementations had a significant bug in this area that was not detected for five years.

Second, it only protects the header extension values, and not their ids or lengths. It also does not protect the CSRCs. As noted above,

this leaves a fair amount of potentially sensitive information exposed.

Third, it bloats the header extension space. Because each extension must be offered in both unencrypted and encrypted forms, twice as many header extensions must be offered, which will in many cases push implementations past the 14-extension limit for the use of one-byte extension headers defined in [\[RFC8285\]](#). Accordingly, implementations will need to use two-byte headers in many cases, which are not supported well by some existing implementations.

Finally, the header extension bloat combined with the need for backwards compatibility results in additional wire overhead. Because two-byte extension headers may not be handled well by existing implementations, one-byte extension identifiers will need to be used for the unencrypted (backwards compatible) forms, and two-byte for the encrypted forms. Thus, deployment of [\[RFC6904\]](#) encryption for header extensions will typically result in multiple extra bytes in each RTP packet, compared to the present situation.

### 1.3. Goals

From the previous analysis, the desired properties of a solution are:

- \*Build on existing [\[RFC3711\]](#) SRTP framework (simple to understand)
- \*Build on existing [\[RFC8285\]](#) header extension framework (simple to implement)
- \*Protection of header extension ids, lengths, and values
- \*Protection of CSRCs when present
- \*Simple signaling
- \*Simple crypto transform and SRTP interactions
- \*Backward compatible with unencrypted endpoints, if desired
- \*Backward compatible with existing RTP tooling

The last point deserves further discussion. While considering possible solutions that would have encrypted more of the RTP header (e.g., the number of CSRCs), lack of support on current tools was inevitable and the additional complexity outweighed the slight improvement in confidentiality by fixing previous solutions. Hence, a new approach was needed to solve the described problem in [Section 1.1](#).

## 2. Terminology

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

## 3. Design

This specification proposes a mechanism to negotiate encryption of all RTP header extensions (ids, lengths, and values) as well as CSRC values. It reuses the existing SRTP framework, is accordingly simple to implement, and is backward compatible with existing RTP packet parsing code, even when support for the mechanism has been negotiated.

Except when explicitly stated otherwise, Cryptex reuses all the framework procedures, transforms and considerations described in [[RFC3711](#)].

## 4. SDP Considerations

Cryptex support is indicated via a new "a=cryptex" SDP attribute defined in this specification.

The new "a=cryptex" attribute is a property attribute as defined in [[RFC8866](#)] section 5.13 and therefore takes no value, and can be used at the session level or media level.

The presence of the "a=cryptex" attribute in the SDP (either in an offer or answer) indicates that the endpoint is capable of receiving RTP packets encrypted with Cryptex, as defined below.

Once each peer has verified that the other party supports receiving RTP packets encrypted with Cryptex, senders can unilaterally decide whether to use or not the Cryptex mechanism on a per packet basis.

If BUNDLE is in use as per [[RFC9143](#)] and the "a=cryptex" attribute is present for a media line, it **MUST** be present for all RTP-based "m=" sections belonging to the same bundle group. This ensures that the encrypted MID header extensions can be processed, allowing to associate RTP streams with the correct "m=" section in each BUNDLE group as specified in [[RFC9143](#)] section 9.2. When used with BUNDLE, this attribute is assigned to the TRANSPORT category [[RFC8859](#)].

Both endpoints can change the Cryptex support status by modifying the session as specified in [[RFC3264](#)] section 8. Generating subsequent SDP offers and answers **MUST** use the same procedures for

including the "a=cryptex" attribute as the ones on the initial offer and answer.

## 5. RTP Header Processing

A General Mechanism for RTP Header Extensions [[RFC8285](#)] defines two values for the "defined by profile" field for carrying one-byte and two-byte header extensions. In order to allow a receiver to determine if an incoming RTP packet is using the encryption scheme in this specification, two new values are defined:

\*0xC0DE for the encrypted version of the one-byte header extensions (instead of 0xBEDE).

\*0xC2DE for the encrypted versions of the two-byte header extensions (instead of 0x100).

In the case of using two-byte header extensions, the extension id with value 256 **MUST NOT** be negotiated, as the value of this id is meant to be contained in the "appbits" of the "defined by profile" field, which are not available when using the values above.

Note that as per [[RFC8285](#)] it is not possible to mix one-byte and two-byte headers on the same RTP packet. Mixing one-byte and two-byte headers on the same RTP stream requires negotiation of the "extmap-allow-mixed" SDP attribute as defined in [[RFC8285](#)] section 4.1.2.

Peers **MAY** negotiate both Cryptex and the Encryption of Header Extensions mechanism defined in [[RFC6904](#)] via SDP offer/answer as described in [Section 4](#), and if both mechanisms are supported, either one can be used for any given packet. However, if a packet is encrypted with Cryptex, it **MUST NOT** also use [[RFC6904](#)] header extension encryption, and vice versa.

If one of the peers has advertised both the ability to receive cryptex and the ability to receive header extensions encrypted as per [[RFC6904](#)] in the SDP exchange, it is **RECOMMENDED** for the other peer to use Cryptex rather than [[RFC6904](#)] when sending RTP packets so all the header extensions and CSRCs are encrypted unless there is a compelling reason to use [[RFC6904](#)] (e.g. a need for some header extensions to be sent in the clear so that so they are processable by RTP middleboxes) in which case, it **SHOULD** use [[RFC6904](#)] instead.

### 5.1. Sending

When the mechanism defined by this specification has been negotiated, sending an RTP packet that has any CSRCs or contains any [[RFC8285](#)] header extensions follows the steps below. This mechanism

**MUST NOT** be used with header extensions other than the [[RFC8285](#)] variety.

If the RTP packet contains one-byte headers, the 16-bit RTP header extension tag **MUST** be set to 0xC0DE to indicate that the encryption has been applied, and the one-byte framing is being used. Otherwise, the header extension tag **MUST** be set to 0xC2DE to indicate encryption has been applied, and the two-byte framing is being used.

If the packet contains CSRCs but no header extensions, an empty extension block consisting of the 0xC0DE tag and a 16-bit length field set to zero (explicitly permitted by [[RFC3550](#)]) **MUST** be appended, and the X bit **MUST** be set to 1 to indicate an extension block is present. This is necessary to provide the receiver an indication that the CSRCs in the packet are encrypted.

The RTP packet **MUST** then be encrypted as described in Encryption Procedure.

## 5.2. Receiving

When receiving an RTP packet that contains header extensions, the "defined by profile" field **MUST** be checked to ensure the payload is formatted according to this specification. If the field does not match one of the values defined above, the implementation **MUST** instead handle it according to the specification that defines that value.

Alternatively, if the implementation considers the use of this specification mandatory and the "defined by profile" field does not match one of the values defined above, it **MUST** stop the processing of the RTP packet and report an error for the RTP stream.

If the RTP packet passes this check, it is then decrypted according to Decryption Procedure, and passed to the next layer to process the packet and its extensions. In the event that a zero-length extension block was added as indicated above, it can be left as-is and will be processed normally.

## 6. Encryption and Decryption

### 6.1. Packet Structure

When this mechanism is active, the SRTP packet is protected as follows:

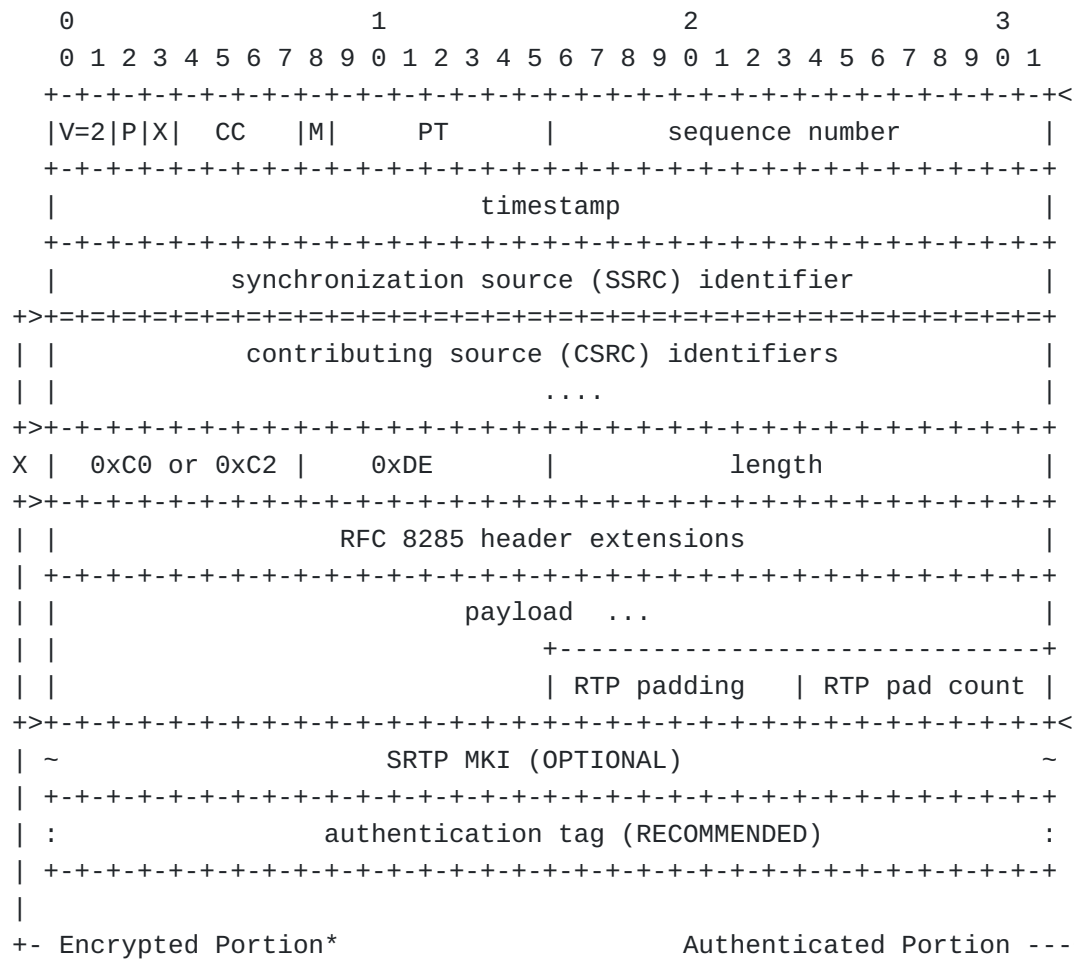


Figure 1

\*Note that, as required by [RFC8285], the 4 bytes at the start of the extension block are not encrypted.

Specifically, the encrypted portion **MUST** include any CSRC identifiers, any RTP header extension (except for the first 4 bytes), and the RTP payload.

## 6.2. Encryption Procedure

The encryption procedure is identical to that of [RFC3711] except for the Encrypted Portion of the SRTP packet. The plaintext input to the cipher is as follows:

```
Plaintext = CSRC identifiers (if used) || header extension data ||
           RTP payload || RTP padding (if used) || RTP pad count (if used).
```

Here "header extension data" refers to the content of the RTP extension field, excluding the first four bytes (the RFC 8285 extension header). The first 4 \* CSRC count (CC) bytes of the ciphertext are placed in the CSRC field of the RTP header. The



remainder of the ciphertext is the RTP payload of the encrypted packet.

To minimize changes to surrounding code, the encryption mechanism can choose to replace a "defined by profile" field from [[RFC8285](#)] with its counterpart defined in RTP Header Processing above and encrypt at the same time.

For AEAD ciphers (e.g., GCM), the 12-byte fixed header and the four-byte header extension header (the "defined by profile" field and the length) are considered AAD, even though they are non-contiguous in the packet if CSRCs are present.

Associated Data: fixed header || extension header (if X=1)

Here "fixed header" refers to the 12-byte fixed portion of the RTP header, and "extension header" refers to the four-byte RFC 8285 extension header ("defined by profile" and extension length).

Implementations can rearrange a packet so that the AAD and plaintext are contiguous by swapping the order of the extension header and the CSRC identifiers, resulting in an intermediate representation of the form shown in [Figure 2](#). After encryption, the CSRCs (now encrypted) and extension header would need to be swapped back to their original positions. A similar operation can be done when decrypting to create contiguous ciphertext and AAD inputs.

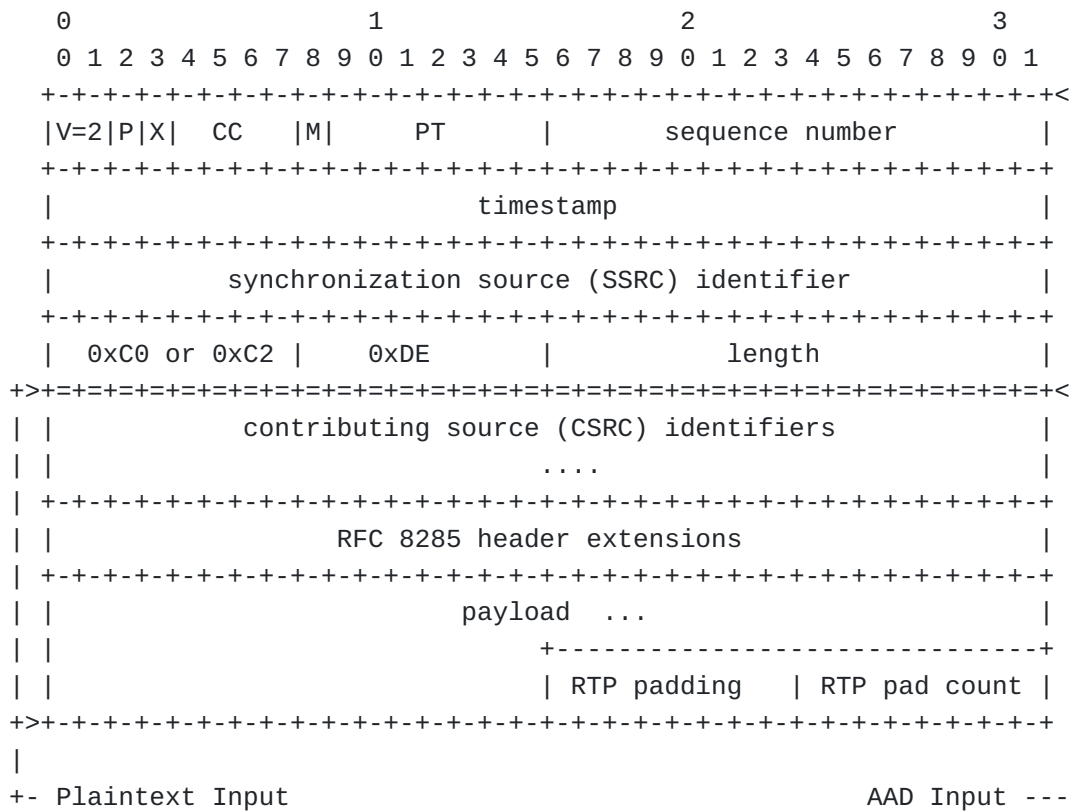


Figure 2: An RTP packet transformed to make Cryptex cipher inputs contiguous

Note: This intermediate representation is only displayed as reference for implementations and is not meant to be sent on the wire.

### 6.3. Decryption Procedure

The decryption procedure is identical to that of [\[RFC3711\]](#) except for the Encrypted Portion of the SRTP packet, which is as shown in the section above.

To minimize changes to surrounding code, the decryption mechanism can choose to replace the "defined by profile" field with its no-encryption counterpart from [\[RFC8285\]](#) and decrypt at the same time.

### 7. Backwards Compatibility

This specification attempts to encrypt as much as possible without interfering with backwards compatibility for systems that expect a certain structure from an RTPv2 packet, including systems that perform demultiplexing based on packet headers. Accordingly, the first two bytes of the RTP packet are not encrypted.

This specification also attempts to reuse the key scheduling from SRTP, which depends on the RTP packet sequence number and SSRC identifier. Accordingly, these values are also not encrypted.

## 8. Security Considerations

All security considerations in [[RFC3711](#)] section 9 are applicable to this specification, except section 9.4. Confidentiality of the RTP Header which is the purpose of this specification.

The risks of using weak or NULL authentication with SRTP, described in Section 9.5 of [[RFC3711](#)], apply to encrypted header extensions as well.

This specification extends SRTP by expanding the Encrypted Portion of the RTP packet, as shown in Packet Structure. It does not change how SRTP authentication works in any way. Given that more of the packet is being encrypted than before, this is necessarily an improvement.

The RTP fields that are left unencrypted (see rationale above) are as follows:

- \*RTP version
- \*padding bit
- \*extension bit
- \*number of CSRCs
- \*marker bit
- \*payload type
- \*sequence number
- \*timestamp
- \*SSRC identifier
- \*number of [[RFC8285](#)] header extensions

These values contain a fixed set (i.e., one that won't be changed by extensions) of information that, at present, is observed to have low sensitivity. In the event any of these values need to be encrypted, SRTP is likely the wrong protocol to use and a fully-encapsulating protocol such as DTLS is preferred (with its attendant per-packet overhead).

## 9. IANA Considerations

### 9.1. SDP cryptex Attribute

This document updates the "Session Description Protocol Parameters" as specified in Section 8.2.4 of [RFC8866]. Specifically, it adds the SDP "a=cryptex" attribute to the Attribute Names (<attribute-name>) registry for both media and session level usage.

Contact name: IETF AVT Working Group or IESG if AVT is closed

Contact email address: avt@ietf.org

Attribute name: cryptex

Attribute syntax: This attribute takes no values.

Attribute semantics: N/A

Attribute value: N/A

Usage level: session, media

Charset dependent: No

Purpose: The presence of this attribute in the SDP indicates that the endpoint is capable of receiving RTP packets encrypted with Cryptex as described in this document.

O/A procedures: SDP O/A procedures are described in Section 4 of this document.

Mux Category: TRANSPORT

## 10. Acknowledgements

The authors wish to thank Lennart Grahl for pointing out many of the issues with the existing header encryption mechanism, as well as suggestions for this proposal. Thanks also to Jonathan Lennox, Inaki Castillo, and Bernard Aboba for their review and suggestions.

## 11. References

### 11.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<https://www.rfc-editor.org/info/rfc3264>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://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, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <<https://www.rfc-editor.org/info/rfc8285>>.
- [RFC8859] Nandakumar, S., "A Framework for Session Description Protocol (SDP) Attributes When Multiplexing", RFC 8859, DOI 10.17487/RFC8859, January 2021, <<https://www.rfc-editor.org/info/rfc8859>>.
- [RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <<https://www.rfc-editor.org/info/rfc8866>>.
- [RFC9143] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 9143, DOI 10.17487/RFC9143, February 2022, <<https://www.rfc-editor.org/info/rfc9143>>.

## 11.2. Informative References

- [RFC6464] Lennox, J., Ed., Iovov, 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, <<https://www.rfc-editor.org/info/rfc6464>>.

[RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <<https://www.rfc-editor.org/info/rfc6465>>.

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

[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, <<https://www.rfc-editor.org/info/rfc7714>>.

## Appendix A. Test Vectors

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

### A.1. AES-CTR

The following section list the test vectors for using cryptex with AES-CTR as per [[RFC3711](#)]

Common values are organized as follows:

Rollover Counter:	00000000
Master Key:	e1f97a0d3e018be0d64fa32c06de4139
Master Salt:	0ec675ad498afeebb6960b3aabe6
Crypto Suite:	AES_CM_128_HMAC_SHA1_80
Session Key:	c61e7a93744f39ee10734afe3ff7a087
Session Salt:	30cbbc08863d8c85d49db34a9ae1
Authentication Key:	cebe321f6fff7716b6fd4ab49af256a156d38baa4

#### A.1.1. RTP Packet with 1-byte header extension

RTP Packet:

900f1235  
decafbad  
cafebabe  
bede0001  
51000200  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

900f1235  
decafbad  
cafebabe  
c0de0001  
eb923652  
51c3e036  
f8de27e9  
c27ee3e0  
b4651d9f  
bc4218a7  
0244522f  
34a5

#### **A.1.1.2. RTP Packet with 2-byte header extension**

RTP Packet:

900f1236  
decafbad  
cafebabe  
10000001  
05020002  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

900f1236  
decafbad  
cafebabe  
c2de0001  
4ed9cc4e  
6a712b30  
96c5ca77  
339d4204  
ce0d7739  
6cab6958  
5fbce381  
94a5

#### **A.1.3. RTP Packet with 1-byte header extension and CSRC fields**

RTP Packet:

920f1238  
decafbad  
cafebabe  
0001e240  
0000b26e  
bede0001  
51000200  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

920f1238  
decafbad  
cafebabe  
8bb6e12b  
5cff16dd  
c0de0001  
92838c8c  
09e58393  
e1de3a9a  
74734d67  
45671338  
c3acf11d  
a2df8423  
bee0

#### **A.1.4. RTP Packet with 2-byte header extension and CSRC fields**

RTP Packet:



920f1239  
decafbad  
cafebabe  
0001e240  
0000b26e  
10000001  
05020002  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

920f1239  
decafbad  
cafebabe  
f70e513e  
b90b9b25  
c2de0001  
bbed4848  
faa64466  
5f3d7f34  
125914e9  
f4d0ae92  
3c6f479b  
95a0f7b5  
3133

#### **A.1.5. RTP Packet with empty 1-byte header extension and CSRC fields**

RTP Packet:

920f123a  
decafbad  
cafebabe  
0001e240  
0000b26e  
bede0000  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

920f123a  
decafbad  
cafebabe  
7130b6ab  
fe2ab0e3  
c0de0000  
e3d9f64b  
25c9e74c  
b4cf8e43  
fb92e378  
1c2c0cea  
b6b3a499  
a14c

#### **A.1.6. RTP Packet with empty 2-byte header extension and CSRC fields**

RTP Packet:

920f123b  
decafbad  
cafebabe  
0001e240  
0000b26e  
10000000  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

920f123b  
decafbad  
cafebabe  
cbf24c12  
4330e1c8  
c2de0000  
599dd45b  
c9d687b6  
03e8b59d  
771fd38e  
88b170e0  
cd31e125  
eabe

#### **A.2. AES-GCM**

The following section list the test vectors for using cryptex with AES-GCM as per [[RFC7714](#)]

Common values are organized as follows:

Rollover Counter:	00000000
Master Key:	000102030405060708090a0b0c0d0e0f
Master Salt:	a0a1a2a3a4a5a6a7a8a9aaab
Crypto Suite:	AEAD_AES_128_GCM
Session Key:	077c6143cb221bc355ff23d5f984a16e
Session Salt:	9af3e95364ebac9c99c5a7c4

#### **A.2.1. RTP Packet with 1-byte header extension**

RTP Packet:

900f1235  
decafbad  
cafebabe  
bede0001  
51000200  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

900f1235  
decafbad  
cafebabe  
c0de0001  
39972dc9  
572c4d99  
e8fc355d  
e743fb2e  
94f9d8ff  
54e72f41  
93bbc5c7  
4ffab0fa  
9fa0fbeb

#### **A.2.2. RTP Packet with 2-byte header extension**

RTP Packet:

900f1236  
decafbad  
cafebabe  
10000001  
05020002  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

900f1236  
decafbad  
cafebabe  
c2de0001  
bb75a4c5  
45cd1f41  
3bdb7daa  
2b1e3263  
de313667  
c9632490  
81b35a65  
f5cb6c88  
b394235f

### **A.2.3. RTP Packet with 1-byte header extension and CSRC fields**

RTP Packet:

920f1238  
decafbad  
cafebabe  
0001e240  
0000b26e  
bede0001  
51000200  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet:

920f1238  
decafbad  
cafebabe  
63bbccc4  
a7f695c4  
c0de0001  
8ad7c71f  
ac70a80c  
92866b4c  
6ba98546  
ef913586  
e95ffaaf  
fe956885  
bb0647a8  
bc094ac8

#### A.2.4. RTP Packet with 2-byte header extension and CSRC fields

RTP Packet :

920f1239  
decafbad  
cafebabe  
0001e240  
0000b26e  
10000001  
05020002  
abababab  
abababab  
abababab  
abababab

Encrypted RTP Packet :

920f1239  
decafbad  
cafebabe  
3680524f  
8d312b00  
c2de0001  
c78d1200  
38422bc1  
11a7187a  
18246f98  
0c059cc6  
bc9df8b6  
26394eca  
344e4b05  
d80fea83

### A.2.5. RTP Packet with empty 1-byte header extension and CSRC fields

RTP Packet:

```
920f123a
decafbad
cafebabe
0001e240
0000b26e
bede0000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

```
920f123a
decafbad
cafebabe
15b6bb43
37906fff
c0de0000
b7b96453
7a2b03ab
7ba5389c
e9331712
6b5d974d
f30c6884
dcb651c5
e120c1da
```

### A.2.6. RTP Packet with empty 2-byte header extension and CSRC fields

RTP Packet:

```
920f123b
decafbad
cafebabe
0001e240
0000b26e
10000000
abababab
abababab
abababab
abababab
```

Encrypted RTP Packet:

920f123b  
decafbad  
cafebabe  
dcb38c9e  
48bf95f4  
c2de0000  
61ee432c  
f9203170  
76613258  
d3ce4236  
c06ac429  
681ad084  
13512dc9  
8b5207d8

### **Authors' Addresses**

Justin Uberti  
Clubhouse

Email: [justin@uberti.name](mailto:justin@uberti.name)

Cullen Jennings  
Cisco

Email: [fluffy@iii.ca](mailto:fluffy@iii.ca)

Sergio Garcia Murillo  
Millicast

Email: [sergio.garcia.murillo@cosmosoftware.io](mailto:sergio.garcia.murillo@cosmosoftware.io)