Internet Engineering Task Force (IETF)
Request for Comments: 7714
Category: Standards Track
ISSN: 2070-1721

D. McGrew
Cisco Systems, Inc.
K. Igoe
National Security Agency

December 2015

AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)

Abstract

This document defines how the AES-GCM Authenticated Encryption with Associated Data family of algorithms can be used to provide confidentiality and data authentication in the Secure Real-time Transport Protocol (SRTP).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7714.

Table of Contents

1. Introduction

The Secure Real-time Transport Protocol (SRTP) [RFC3711] is a profile of the Real-time Transport Protocol (RTP) [RFC3550], which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the control traffic for RTP, the Real-time Transport Control Protocol (RTCP). It is important to note that the outgoing SRTP packets from a single endpoint may be originating from several independent data sources.

Authenticated Encryption [BN00] is a form of encryption that, in addition to providing confidentiality for the Plaintext that is encrypted, provides a way to check its integrity and authenticity. Authenticated Encryption with Associated Data, or AEAD [R02], adds the ability to check the integrity and authenticity of some Associated Data (AD), also called "Additional Authenticated Data" (AAD), that is not encrypted. This specification makes use of the interface to a generic AEAD algorithm as defined in [RFC5116].

The Advanced Encryption Standard (AES) is a block cipher that provides a high level of security and can accept different key sizes. AES Galois/Counter Mode (AES-GCM) [GCM] is a family of AEAD algorithms based upon AES. This specification makes use of the AES versions that use 128-bit and 256-bit keys, which we call "AES-128" and "AES-256", respectively.

Any AEAD algorithm provides an intrinsic authentication tag. In many applications, the authentication tag is truncated to less than full length. In this specification, the authentication tag MUST NOT be truncated. The authentications tags MUST be a full 16 octets in length. When used in SRTP/SRTCP, AES-GCM will have two configurations:

AEAD_AES_128_GCM	AES-128 with a 16-octet authentication tag
AEAD_AES_256_GCM	AES-256 with a 16-octet authentication tag

The key size is set when the session is initiated and SHOULD NOT be altered.

The Galois/Counter Mode of operation (GCM) is an AEAD mode of operation for block ciphers. GCM uses Counter Mode to encrypt the data, an operation that can be efficiently pipelined. Further, GCM authentication uses operations that are particularly well suited to efficient implementation in hardware, making it especially appealing for high-speed implementations, or for implementations in an efficient and compact circuit.

In summary, this document defines how to use an AEAD algorithm, particularly AES-GCM, to provide confidentiality and message authentication within SRTP and SRTCP packets.

2. Conventions Used in This Document

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 [RFC2119].

3. Overview of the SRTP/SRTCP AEAD Security Architecture

SRTP/SRTCP AEAD security is based upon the following principles:

a) Both privacy and authentication are based upon the use of symmetric algorithms. An AEAD algorithm such as AES-GCM combines privacy and authentication into a single process.

b) A secret master key is shared by all participating endpoints -- both those originating SRTP/SRTCP packets and those receiving these packets. Any given master key MAY be used simultaneously by several endpoints to originate SRTP/SRTCP packets (as well as one or more endpoints using this master key to process inbound data).

c) A Key Derivation Function (KDF) is applied to the shared master key value to form separate encryption keys, authentication keys, and salting keys for SRTP and for SRTCP (a total of six keys). This process is described in Section 4.3 of [RFC3711]. The master key MUST be at least as large as the encryption key derived from it. Since AEAD algorithms such as AES-GCM combine encryption and authentication into a single process, AEAD algorithms do not make use of separate authentication keys.

d) Aside from making modifications to IANA registries to allow AES-GCM to work with Security Descriptions (SDES), Datagram Transport Layer Security for Secure RTP (DTLS-SRTP), and Multimedia Internet KEYing (MIKEY), the details of how the master key is established and shared between the participants are outside the scope of this document. Similarly, any mechanism for rekeying an existing session is outside the scope of the document.

e) Each time an instantiation of AES-GCM is invoked to encrypt and authenticate an SRTP or SRTCP data packet, a new Initialization Vector (IV) is used. SRTP combines the 4-octet Synchronization Source (SSRC) identifier, the 4-octet Rollover Counter (ROC), and the 2-octet Sequence Number (SEQ) with the 12-octet encryption salt to form a 12-octet IV (see Section 8.1). SRTCP combines the SSRC and 31-bit SRTCP index with the encryption salt to form a 12-octet IV (see Section 9.1).

4. Terminology

The following terms have very specific meanings in the context of this RFC:

Instantiation: In AEAD, an instantiation is an (Encryption_key, salt) pair together with all of the data structures (for example, counters) needed for it to function properly. In SRTP/SRTCP, each endpoint will need two instantiations of the AEAD algorithm for each master key in its possession: one instantiation for SRTP traffic and one instantiation for SRTCP traffic.

Invocation: SRTP/SRTCP data streams are broken into packets. Each packet is processed by a single invocation of the appropriate instantiation of the AEAD algorithm.

In many applications, each endpoint will have one master key for processing outbound data but may have one or more separate master keys for processing inbound data.

5. Generic AEAD Processing

5.1. Types of Input Data

Associated Data: Data that is to be authenticated but not encrypted.

Plaintext: Data that is to be both encrypted and authenticated.

Raw Data: Data that is to be neither encrypted nor authenticated.

Which portions of SRTP/SRTCP packets that are to be treated as Associated Data, which are to be treated as Plaintext, and which are to be treated as Raw Data are covered in Sections 8.2, 9.2, and 9.3.

5.2. AEAD Invocation Inputs and Outputs

5.2.1. Encrypt Mode

Inputs:

Encryption_key	Octet string, either 16 or 32 octets long
Initialization_Vector	Octet string, 12 octets long
Associated_Data	Octet string of variable length
Plaintext	Octet string of variable length

Outputs:

Ciphertext*	Octet string, length = length(Plaintext) + tag_length

(*): In AEAD, the authentication tag in embedded in the ciphertext. When GCM is being used, the ciphertext consists of the encrypted Plaintext followed by the authentication tag.

5.2.2. Decrypt Mode

Inputs:	
Encryption_key	Octet string, either 16 or 32 octets long
Initialization_Vector	Octet string, 12 octets long
Associated_Data	Octet string of variable length
Ciphertext	Octet string of variable length
Outputs:	
Plaintext	Octet string, length = length(Ciphertext) - tag_length
Validity_Flag	Boolean, TRUE if valid, FALSE otherwise

5.3. Handling of AEAD Authentication

AEAD requires that all incoming packets MUST pass AEAD authentication before any other action takes place. Plaintext and Associated Data MUST NOT be released until the AEAD authentication tag has been validated. Further, the ciphertext MUST NOT be decrypted until the AEAD tag has been validated.

Should the AEAD tag prove to be invalid, the packet in question is to be discarded and a Validation Error flag raised. Local policy determines how this flag is to be handled and is outside the scope of this document.

6. Counter Mode Encryption

Each outbound packet uses a 12-octet IV and an encryption key to form two outputs:

- a 16-octet first_key_block, which is used in forming the authentication tag, and
- a keystream of octets, formed in blocks of 16 octets each

The first 16-octet block of the key is saved for use in forming the authentication tag, and the remainder of the keystream is XORed to the Plaintext to form the cipher. This keystream is formed one block at a time by inputting the concatenation of a 12-octet IV (see Sections 8.1 and 9.1) with a 4-octet block to AES. The pseudocode below illustrates this process:

```
 def GCM_keystream( Plaintext_len, IV, Encryption_key ):
     assert Plaintext_len <= (2**36) - 32 ## measured in octets
     key_stream = ""
     block_counter = 1
     first_key_block = AES_ENC( data=IV||block_counter,
                                key=Encryption_key )
     while len(key_stream) < Plaintext_len:
         block_counter = block_counter + 1
         key_block = AES_ENC( data=IV||block_counter,
                              key=Encryption_key )
         key_stream = key_stream||key_block
     key_stream = truncate( key_stream, Plaintext_len )
     return( first_key_block, key_stream )
```

In theory, this keystream generation process allows for the encryption of up to (2^36) - 32 octets per invocation (i.e., per packet), far longer than is actually required.

With any counter mode, if the same (IV, Encryption_key) pair is used twice, precisely the same keystream is formed. As explained in Section 9.1 of [RFC3711], this is a cryptographic disaster. For GCM, the consequences are even worse, since such a reuse compromises GCM's integrity mechanism not only for the current packet stream but for all future uses of the current encryption_key.

7. Unneeded SRTP/SRTCP Fields

AEAD Counter Mode encryption removes the need for certain existing SRTP/SRTCP mechanisms.

7.1. SRTP/SRTCP Authentication Tag Field

The AEAD message authentication mechanism MUST be the primary message authentication mechanism for AEAD SRTP/SRTCP. Additional SRTP/SRTCP authentication mechanisms SHOULD NOT be used with any AEAD algorithm, and the optional SRTP/SRTCP authentication tags are NOT RECOMMENDED and SHOULD NOT be present. Note that this contradicts Section 3.4 of [RFC3711], which makes the use of the SRTCP authentication tag field mandatory, but the presence of the AEAD authentication renders the older authentication methods redundant.

Rationale: Some applications use the SRTP/SRTCP authentication tag as a means of conveying additional information, notably [RFC4771]. This document retains the authentication tag field primarily to preserve compatibility with these applications.

7.2. RTP Padding

AES-GCM does not require that the data be padded out to a specific block size, reducing the need to use the padding mechanism provided by RTP. It is RECOMMENDED that the RTP padding mechanism not be used unless it is necessary to disguise the length of the underlying Plaintext.

8. AES-GCM Processing for SRTP

8.1. SRTP IV Formation for AES-GCM

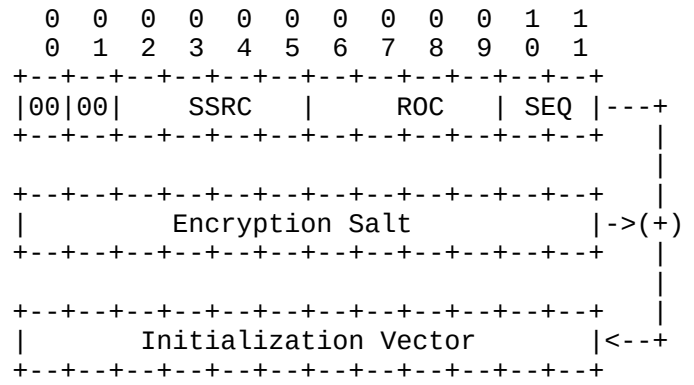


Figure 1: AES-GCM SRTP Initialization Vector Formation

The 12-octet IV used by AES-GCM SRTP is formed by first concatenating 2 octets of zeroes, the 4-octet SSRC, the 4-octet rollover counter (ROC), and the 2-octet sequence number (SEQ). The resulting 12-octet value is then XORed to the 12-octet salt to form the 12-octet IV.

8.2. Data Types in SRTP Packets

All SRTP packets MUST be both authenticated and encrypted. The data fields within the RTP packets are broken into Associated Data, Plaintext, and Raw Data, as follows (see Figure 2):

Associated Data: The version V (2 bits), padding flag P (1 bit), extension flag X (1 bit), Contributing Source (CSRC) count CC (4 bits), marker M (1 bit), Payload Type PT (7 bits), sequence number (16 bits), timestamp (32 bits), SSRC (32 bits), optional CSRC identifiers (32 bits each), and optional RTP extension (variable length).

Plaintext: The RTP payload (variable length), RTP padding (if used, variable length), and RTP pad count (if used, 1 octet).

Raw Data: The optional variable-length SRTP Master Key Identifier (MKI) and SRTP authentication tag (whose use is NOT RECOMMENDED). These fields are appended after encryption has been performed.

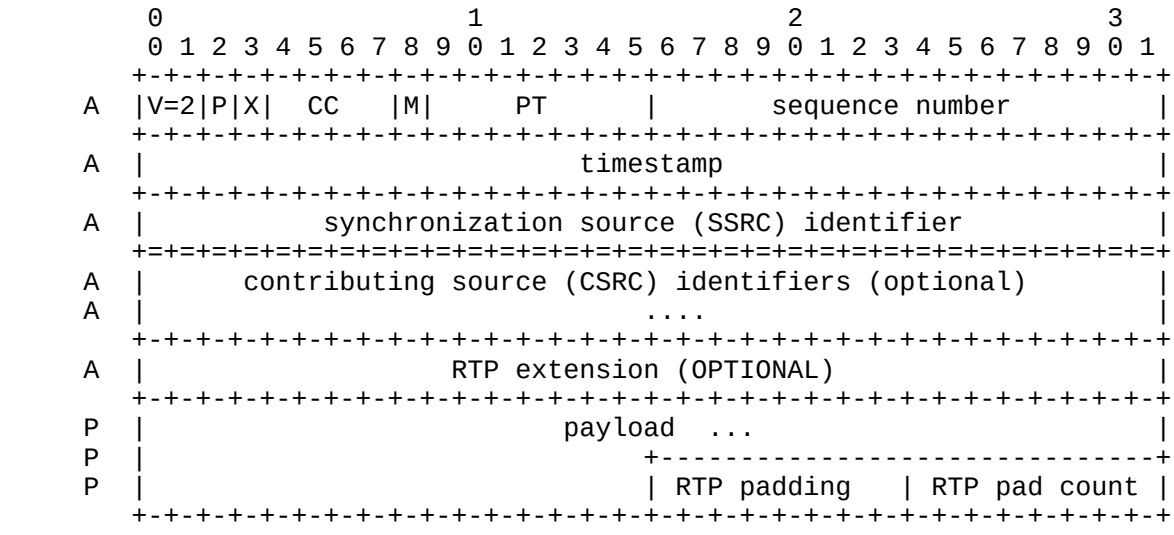


Figure 2: Structure of an RTP Packet before Authenticated Encryption

Since the AEAD ciphertext is larger than the Plaintext by exactly the length of the AEAD authentication tag, the corresponding SRTP-encrypted packet replaces the Plaintext field with a slightly larger field containing the cipher. Even if the Plaintext field is empty, AEAD encryption must still be performed, with the resulting cipher consisting solely of the authentication tag. This tag is to be placed immediately before the optional variable-length SRTP MKI and SRTP authentication tag fields.

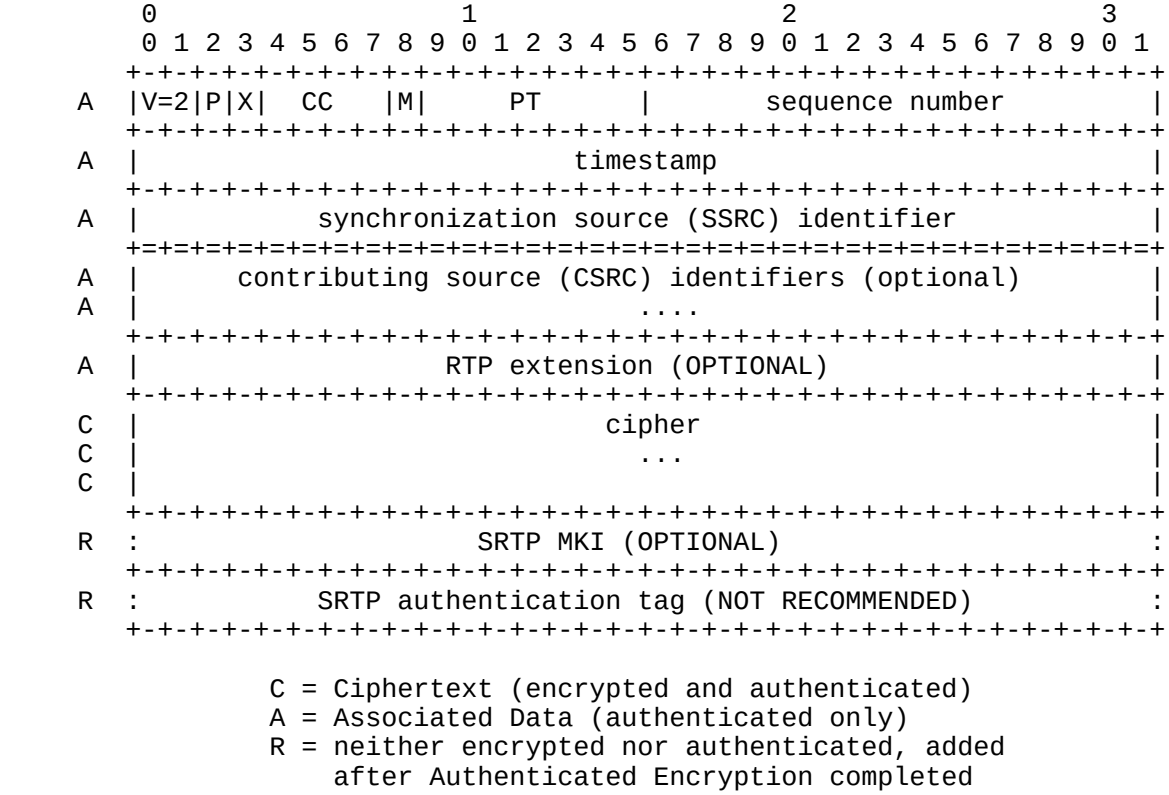


Figure 3: Structure of an SRTP Packet after Authenticated Encryption

8.3. Handling Header Extensions

RTP header extensions were first defined in [RFC3550]. [RFC6904] describes how these header extensions are to be encrypted in SRTP.

When RFC 6904 is in use, a separate keystream is generated to encrypt selected RTP header extension elements. For the AEAD_AES_128_GCM algorithm, this keystream MUST be generated in the manner defined in [RFC6904], using the AES Counter Mode (AES-CM) transform. For the

AEAD_AES_256_GCM algorithm, the keystream MUST be generated in the manner defined for the AES_256_CM transform. The originator must perform any required header extension encryption before the AEAD algorithm is invoked.

As with the other fields contained within the RTP header, both encrypted and unencrypted header extensions are to be treated by the AEAD algorithm as Associated Data (AD). Thus, the AEAD algorithm does not provide any additional privacy for the header extensions, but it does provide integrity and authentication.

8.4. Prevention of SRTP IV Reuse

In order to prevent IV reuse, we must ensure that the (ROC,SEQ,SSRC) triple is never used twice with the same master key. The following two scenarios illustrate this issue:

Counter Management: A rekey MUST be performed to establish a new master key before the (ROC,SEQ) pair cycles back to its original value. Note that this scenario implicitly assumes that either (1) the outgoing RTP process is trusted to not attempt to repeat a (ROC,SEQ) value or (2) the encryption process ensures that both the SEQ and ROC numbers of the packets presented to it are always incremented in the proper fashion. This is particularly important for GCM, since using the same (ROC,SEQ) value twice compromises the authentication mechanism. For GCM, the (ROC,SEQ) and SSRC values used MUST be generated or checked by either the SRTP implementation or a module (e.g., the RTP application) that can be considered equally trustworthy. While [RFC3711] allows the detection of SSRC collisions after they happen, SRTP using GCM with shared master keys MUST prevent an SSRC collision from happening even once.

SSRC Management: For a given master key, the set of all SSRC values used with that master key must be partitioned into disjoint pools, one pool for each endpoint using that master key to originate outbound data. Each such originating endpoint MUST only issue SSRC values from the pool it has been assigned. Further, each originating endpoint MUST maintain a history of outbound SSRC

identifiers that it has issued within the lifetime of the current master key, and when a new SSRC requests an SSRC identifier it MUST NOT be given an identifier that has been previously issued. A rekey MUST be performed before any of the originating endpoints using that master key exhaust their pools of SSRC values. Further, the identity of the entity giving out SSRC values MUST be verified, and the SSRC signaling MUST be integrity protected.

9. AES-GCM Processing of SRTCP Compound Packets

All SRTCP compound packets MUST be authenticated, but unlike SRTP, SRTCP packet encryption is optional. A sender can select which packets to encrypt and indicates this choice with a 1-bit Encryption flag (located just before the 31-bit SRTCP index).

9.1. SRTCP IV Formation for AES-GCM

The 12-octet IV used by AES-GCM SRTCP is formed by first concatenating 2 octets of zeroes, the 4-octet SSRC identifier, 2 octets of zeroes, a single "0" bit, and the 31-bit SRTCP index. The resulting 12-octet value is then XORed to the 12-octet salt to form the 12-octet IV.

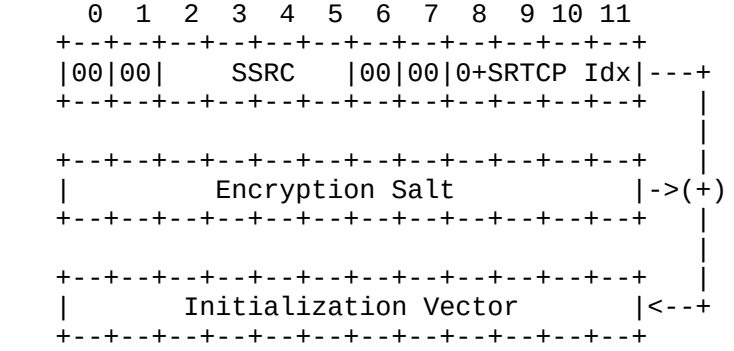
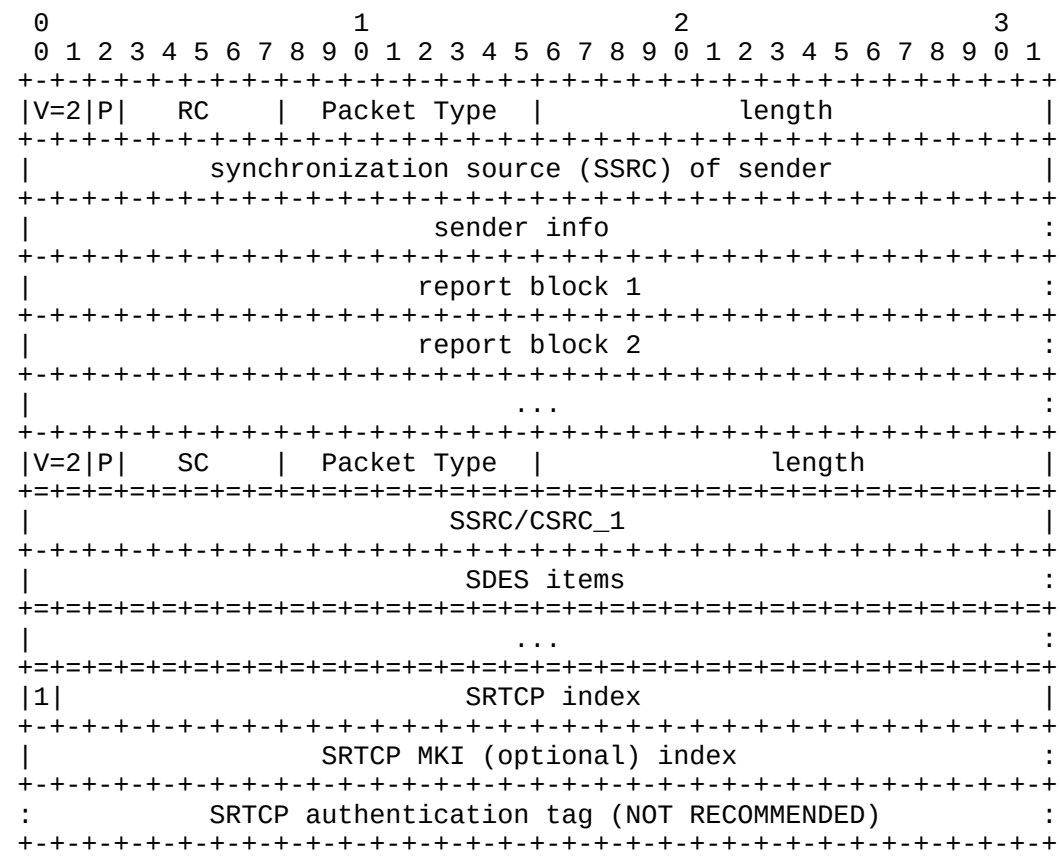


Figure 4: SRTCP Initialization Vector Formation

9.2. Data Types in Encrypted SRTCP Compound Packets



P = Plaintext (to be encrypted and authenticated)
A = Associated Data (to be authenticated only)
R = neither encrypted nor authenticated, added after encryption

Figure 5: AEAD SRTCP Inputs When Encryption Flag = 1
(The fields are defined in RFC 3550.)

When the Encryption flag is set to 1, the SRTCP packet is broken into Plaintext, Associated Data, and Raw (untouched) Data (as shown above in Figure 5):

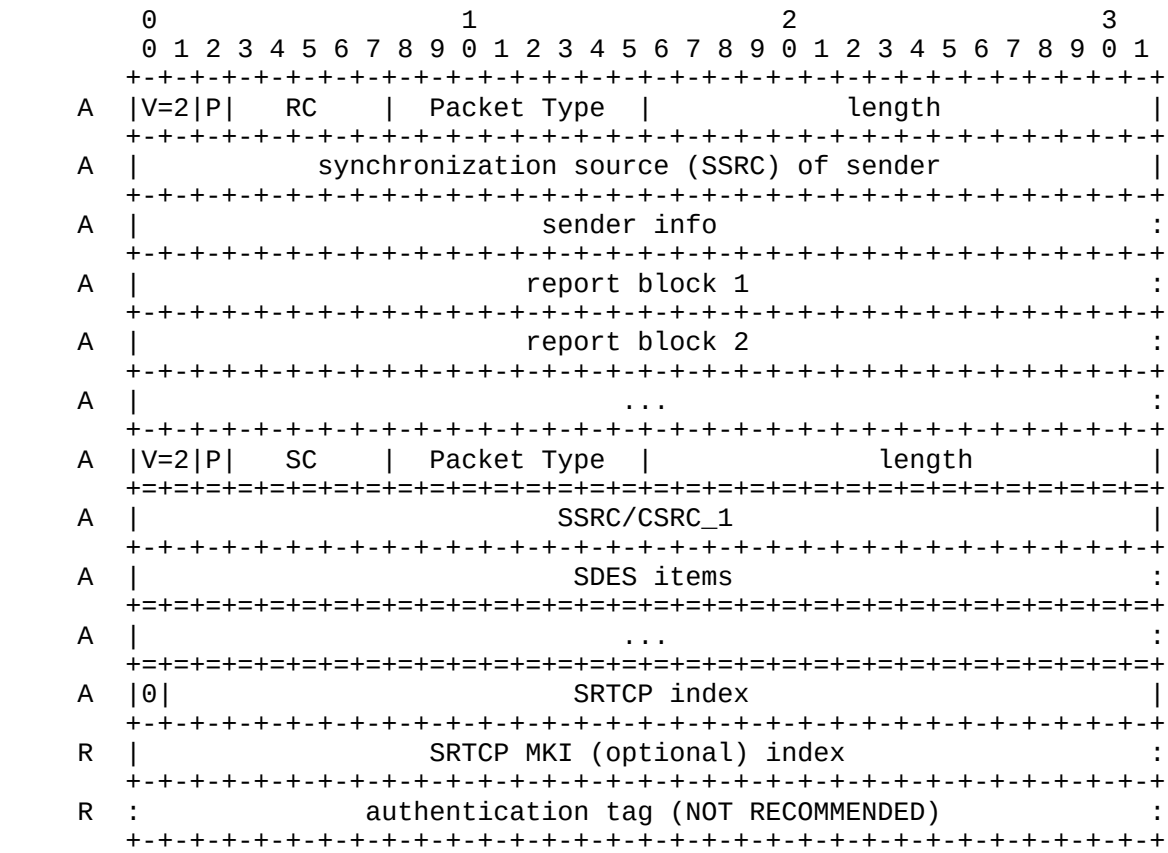
Associated Data: The packet version V (2 bits), padding flag P (1 bit), reception report count RC (5 bits), Packet Type (8 bits), length (2 octets), SSRC (4 octets), Encryption flag (1 bit), and SRTCP index (31 bits).

Raw Data: The optional variable-length SRTCP MKI and SRTCP authentication tag (whose use is NOT RECOMMENDED).

Plaintext: All other data.

Note that the Plaintext comes in one contiguous field. Since the AEAD cipher is larger than the Plaintext by exactly the length of the AEAD authentication tag, the corresponding SRTCP-encrypted packet replaces the Plaintext field with a slightly larger field containing the cipher. Even if the Plaintext field is empty, AEAD encryption must still be performed, with the resulting cipher consisting solely of the authentication tag. This tag is to be placed immediately before the Encryption flag and SRTCP index.

9.3. Data Types in Unencrypted SRTCP Compound Packets



A = Associated Data (to be authenticated only)
R = neither encrypted nor authenticated, added after encryption

Figure 6: AEAD SRTCP Inputs When Encryption Flag = 0

When the Encryption flag is set to 0, the SRTCP compound packet is broken into Plaintext, Associated Data, and Raw (untouched) Data, as follows (see Figure 6):

Plaintext: None.

Raw Data: The variable-length optional SRTCP MKI and SRTCP authentication tag (whose use is NOT RECOMMENDED).

Associated Data: All other data.

Even though there is no ciphertext in this RTCP packet, AEAD encryption returns a cipher field that is precisely the length of the AEAD authentication tag. This cipher is to be placed before the Encryption flag and the SRTCP index in the authenticated SRTCP packet.

9.4. Prevention of SRTCP IV Reuse

A new master key MUST be established before the 31-bit SRTCP index cycles back to its original value. Ideally, a rekey should be performed and a new master key put in place well before the SRTCP index cycles back to the starting value.

The comments on SSRC management in Section 8.4 also apply.

10. Constraints on AEAD for SRTP and SRTCP

In general, any AEAD algorithm can accept inputs with varying lengths, but each algorithm can accept only a limited range of lengths for a specific parameter. In this section, we describe the constraints on the parameter lengths that any AEAD algorithm must support to be used in AEAD-SRTP. Additionally, we specify a complete parameter set for one specific family of AEAD algorithms, namely AES-GCM.

All AEAD algorithms used with SRTP/SRTCP MUST satisfy the five constraints listed below:

Parameter	Meaning	Value
A_MAX	maximum Associated Data length	MUST be at least 12 octets.
N_MIN	minimum nonce (IV) length	MUST be 12 octets.
N_MAX	maximum nonce (IV) length	MUST be 12 octets.
P_MAX	maximum Plaintext length per invocation	GCM: MUST be <= 2^36 - 32 octets.
C_MAX	maximum ciphertext length per invocation	GCM: MUST be <= 2^36 - 16 octets.

For the sake of clarity, we specify three additional parameters:

AEAD authentication tag length	MUST be 16 octets
Maximum number of invocations for a given instantiation	SRTP: MUST be at most 2^48 SRTCP: MUST be at most 2^31
Block Counter size	GCM: MUST be 32 bits

The reader is reminded that the ciphertext is longer than the Plaintext by exactly the length of the AEAD authentication tag.

11. Key Derivation Functions

A Key Derivation Function (KDF) is used to derive all of the required encryption and authentication keys from a secret value shared by the endpoints. The AEAD_AES_128_GCM algorithm MUST use the (128-bit) AES_CM PRF KDF described in [RFC3711]. AEAD_AES_256_GCM MUST use the AES_256_CM_PRF KDF described in [RFC6188].

12. Summary of AES-GCM in SRTP/SRTCP

For convenience, much of the information about the use of the AES-GCM family of algorithms in SRTP is collected in the tables contained in this section.

The AES-GCM family of AEAD algorithms is built around the AES block cipher algorithm. AES-GCM uses AES-CM for encryption and Galois Message Authentication Code (GMAC) for authentication. A detailed description of the AES-GCM family can be found in [RFC5116]. The following members of the AES-GCM family may be used with SRTP/SRTCP:

Name	Key Size	AEAD Tag Size	Reference
AEAD_AES_128_GCM	16 octets	16 octets	[RFC5116]
AEAD_AES_256_GCM	32 octets	16 octets	[RFC5116]

Table 1: AES-GCM Algorithms for SRTP/SRTCP

Any implementation of AES-GCM SRTP MUST support both AEAD_AES_128_GCM and AEAD_AES_256_GCM. Below, we summarize parameters associated with these two GCM algorithms:

Parameter	Value
Master key length	128 bits
Master salt length	96 bits
Key Derivation Function	AES_CM PRF [RFC3711]
Maximum key lifetime (SRTP)	2^48 packets
Maximum key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	AEAD_AES_128_GCM
AEAD authentication tag length	128 bits

Table 2: The AEAD_AES_128_GCM Crypto Suite

Parameter	Value
Master key length	256 bits
Master salt length	96 bits
Key Derivation Function	AES_256_CM_PRF [RFC6188]
Maximum key lifetime (SRTP)	2^48 packets
Maximum key lifetime (SRTCP)	2^31 packets
Cipher (for SRTP and SRTCP)	AEAD_AES_256_GCM
AEAD authentication tag length	128 bits

Table 3: The AEAD_AES_256_GCM Crypto Suite

13. Security Considerations

13.1. Handling of Security-Critical Parameters

As with any security process, the implementer must take care to ensure that cryptographically sensitive parameters are properly handled. Many of these recommendations hold for all SRTP cryptographic algorithms, but we include them here to emphasize their importance.

- If the master salt is to be kept secret, it MUST be properly erased when no longer needed.

- The secret master key and all keys derived from it MUST be kept secret. All keys MUST be properly erased when no longer needed.

- At the start of each packet, the Block Counter MUST be reset to 1. The Block Counter is incremented after each block key has been produced, but it MUST NOT be allowed to exceed 2^32 - 1 for GCM. Note that even though the Block Counter is reset at the start of each packet, IV uniqueness is ensured by the inclusion of SSRC/ROC/SEQ or the SRTCP index in the IV. (The reader is reminded that the first block of key produced is reserved for use in authenticating the packet and is not used to encrypt Plaintext.)

- Each time a rekey occurs, the initial values of both the 31-bit SRTCP index and the 48-bit SRTP packet index (ROC||SEQ) MUST be saved in order to prevent IV reuse.

- Processing MUST cease if either the 31-bit SRTCP index or the 48-bit SRTP packet index (ROC||SEQ) cycles back to its initial value. Processing MUST NOT resume until a new SRTP/SRTCP session has been established using a new SRTP master key. Ideally, a rekey should be done well before any of these counters cycle.

13.2. Size of the Authentication Tag

We require that the AEAD authentication tag be 16 octets, in order to effectively eliminate the risk of an adversary successfully introducing fraudulent data. Though other protocols may allow the use of truncated authentication tags, the consensus of the authors and the working group is that risks associated with using truncated AES-GCM tags are deemed too high to allow the use of truncated authentication tags in SRTP/SRTCP.

14. IANA Considerations

14.1. SDES

"Session Description Protocol (SDP) Security Descriptions for Media Streams" [RFC4568] defines SRTP "crypto suites". A crypto suite corresponds to a particular AEAD algorithm in SRTP. In order to allow security descriptions to signal the use of the algorithms defined in this document, IANA has registered the following crypto suites in the "SRTP Crypto Suite Registrations" subregistry of the "Session Description Protocol (SDP) Security Descriptions" registry. The ABNF [RFC5234] syntax is as follows:

```
srtp-crypto-suite-ext = "AEAD_AES_128_GCM"    /
                        "AEAD_AES_256_GCM"    /
                        srtp-crypto-suite-ext
```

14.2. DTLS-SRTP

DTLS-SRTP [RFC5764] defines DTLS-SRTP "SRTP protection profiles". These profiles also correspond to the use of an AEAD algorithm in SRTP. In order to allow the use of the algorithms defined in this document in DTLS-SRTP, IANA has registered the following SRTP protection profiles:

```
SRTP_AEAD_AES_128_GCM    = {0x00, 0x07}
SRTP_AEAD_AES_256_GCM    = {0x00, 0x08}
```

Below, we list the SRTP transform parameters for each of these protection profiles. Unless separate parameters for SRTP and SRTCP are explicitly listed, these parameters apply to both SRTP and SRTCP.

```
SRTP_AEAD_AES_128_GCM
     cipher:                 AES_128_GCM
     cipher_key_length:      128 bits
     cipher_salt_length:     96 bits
     aead_auth_tag_length:   16 octets
     auth_function:          NULL
     auth_key_length:        N/A
     auth_tag_length:        N/A
     maximum lifetime:       at most 2^31 SRTCP packets and
                               at most 2^48 SRTP packets

SRTP_AEAD_AES_256_GCM
     cipher:                 AES_256_GCM
     cipher_key_length:      256 bits
     cipher_salt_length:     96 bits
     aead_auth_tag_length:   16 octets
     auth_function:          NULL
     auth_key_length:        N/A
     auth_tag_length:        N/A
     maximum lifetime:       at most 2^31 SRTCP packets and
                               at most 2^48 SRTP packets
```

Note that these SRTP protection profiles do not specify an auth_function, auth_key_length, or auth_tag_length, because all of these profiles use AEAD algorithms and thus 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.

14.3. MIKEY

In accordance with "MIKEY: Multimedia Internet KEYing" [RFC3830], IANA maintains several subregistries under "Multimedia Internet KEYing (MIKEY) Payload Name Spaces". Per this document, additions have been made to two of the MIKEY subregistries.

In the "MIKEY Security Protocol Parameters" subregistry, the following has been added:

Type	Meaning	Possible Values
20	AEAD authentication tag length	16 octets

This list is, of course, intended for use with GCM. It is conceivable that new AEAD algorithms introduced at some point in the future may require a different set of authentication tag lengths.

In the "Encryption algorithm (Value 0)" subregistry (derived from Table 6.10.1.b of [RFC3830]), the following has been added:

SRTP Encr. Algorithm	Value	Default Session Encr. Key Length	Default Auth. Tag Length
AES-GCM	6	16 octets	16 octets

The encryption algorithm, session encryption key length, and AEAD authentication tag sizes received from MIKEY fully determine the AEAD algorithm to be used. The exact mapping is described in Section 15.

15. Parameters for Use with MIKEY

MIKEY specifies the algorithm family separately from the key length (which is specified by the Session Encryption key length) and the authentication tag length (specified by the AEAD authentication tag length).

	Encryption Algorithm	Encryption Key Length	AEAD Auth. Tag Length
AEAD_AES_128_GCM	AES-GCM	16 octets	16 octets
AEAD_AES_256_GCM	AES-GCM	32 octets	16 octets

Table 4: Mapping MIKEY Parameters to AEAD Algorithms

Section 11 of this document restricts the choice of KDF for AEAD algorithms. To enforce this restriction in MIKEY, we require that the SRTP Pseudorandom Function (PRF) has value AES-CM whenever an AEAD algorithm is used. Note that, according to Section 6.10.1 of [RFC3830], the input key length of the KDF (i.e., the SRTP master key length) is always equal to the session encryption key length. This means, for example, that AEAD_AES_256_GCM will use AES_256_CM_PRF as the KDF.

16. Some RTP Test Vectors

The examples in this section are all based upon the same RTP packet

```
8040f17b 8041f8d3 5501a0b2 47616c6c
69612065 7374206f 6d6e6973 20646976
69736120 696e2070 61727465 73207472
6573
```

consisting of a 12-octet header (8040f17b 8041f8d3 5501a0b2) and a 38-octet payload (47616c6c 69612065 7374206f 6d6e6973 20646976 69736120 696e2070 61727465 73207472 6573), which is just the ASCII string "Gallia est omnis divisa in partes tres". The salt used (51756964 2070726f 2071756f) comes from the ASCII string "Quid pro quo". The 16-octet (128-bit) key is 00 01 02 ... 0f, and the 32-octet (256-bit) key is 00 01 02 ... 1f. At the time this document was written, the RTP payload type (1000000 binary = 64 decimal) was an unassigned value.

As shown in Section 8.1, the IV is formed by XORing two 12-octet values. The first 12-octet value is formed by concatenating two zero octets, the 4-octet SSRC (found in the ninth through 12th octets of the packet), the 4-octet rollover counter (ROC) maintained at each end of the link, and the 2-octet sequence number (SEQ) (found in the third and fourth octets of the packet). The second 12-octet value is the salt, a value that is held constant at least until the key is changed.

```
           | Pad |   SSRC    |    ROC    | SEQ |
            00 00 55 01 a0 b2 00 00 00 00 f1 7b
     salt   51 75 69 64 20 70 72 6f 20 71 75 6f
           ------------------------------------
       IV   51 75 3c 65 80 c2 72 6f 20 71 84 14
```

All of the RTP examples use this IV.

16.1. SRTP AEAD_AES_128_GCM

16.1.1. SRTP AEAD_AES_128_GCM Encryption

Encrypting the following packet:

```
     8040f17b 8041f8d3 5501a0b2 47616c6c
     69612065 7374206f 6d6e6973 20646976
     69736120 696e2070 61727465 73207472
     6573
```

Form the IV

```
       | Pad |   SSRC    |    ROC    | SEQ |
        00 00 55 01 a0 b2 00 00 00 00 f1 7b
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
```

```
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
AAD: 8040f17b 8041f8d3 5501a0b2
 PT: 47616c6c 69612065 7374206f 6d6e6973
     20646976 69736120 696e2070 61727465
     73207472 6573
 IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
  H: c6a13b37878f5b826f4f8162a1c8d879
```

Encrypt the Plaintext

```
  block # 0
    IV||blk_cntr: 51753c6580c2726f2071841400000002
       key_block: b5 2c 8f cf 92 55 fe 09 df ce a6 73 f0 10 22 b9
     plain_block: 47 61 6c 6c 69 61 20 65 73 74 20 6f 6d 6e 69 73
    cipher_block: f2 4d e3 a3 fb 34 de 6c ac ba 86 1c 9d 7e 4b ca
  block # 1
    IV||blk_cntr: 51753c6580c2726f2071841400000003
       key_block: 9e 07 52 a3 64 5a 2f 4f 2b cb d4 0a 30 b5 a5 fe
     plain_block: 20 64 69 76 69 73 61 20 69 6e 20 70 61 72 74 65
    cipher_block: be 63 3b d5 0d 29 4e 6f 42 a5 f4 7a 51 c7 d1 9b
  block # 2
    IV||blk_cntr: 51753c6580c2726f2071841400000004
       key_block: 45 fe 4e ad ed 40 0a 5d 1a f3 63 f9 0c e1 49 3b
     plain_block: 73 20 74 72 65 73
    cipher_block: 36 de 3a df 88 33
```

Cipher before tag appended

```
     f24de3a3 fb34de6c acba861c 9d7e4bca
     be633bd5 0d294e6f 42a5f47a 51c7d19b
     36de3adf 8833
```

```
   Compute the GMAC tag

     Process the AAD
           AAD word: 8040f17b8041f8d35501a0b200000000
       partial hash: bcfb3d1d0e6e3e78ba45403377dba11b

     Process the cipher
        cipher word: f24de3a3fb34de6cacba861c9d7e4bca
       partial hash: 0ebc0abe1b15b32fedd2b07888c1ef61
        cipher word: be633bd50d294e6f42a5f47a51c7d19b
       partial hash: 438e5797011ea860585709a2899f4685
        cipher word: 36de3adf883300000000000000000000
       partial hash: 336fb643310d7bac2aeaa76247f6036d

     Process the length word
        length word: 00000000000000600000000000000130
       partial hash: 1b964067078c408c4e442a8f015e5264

   Turn GHASH into GMAC
              GHASH: 1b 96 40 67 07 8c 40 8c 4e 44 2a 8f 01 5e 52 64
                 K0: 92 0b 3f 40 b9 3d 2a 1d 1c 8b 5c d1 e5 67 5e aa
         full GMAC: 89 9d 7f 27 be b1 6a 91 52 cf 76 5e e4 39 0c ce

   Cipher with tag
        f24de3a3 fb34de6c acba861c 9d7e4bca
        be633bd5 0d294e6f 42a5f47a 51c7d19b
        36de3adf 8833899d 7f27beb1 6a9152cf
        765ee439 0cce

   Encrypted and tagged packet:
        8040f17b 8041f8d3 5501a0b2 f24de3a3
        fb34de6c acba861c 9d7e4bca be633bd5
        0d294e6f 42a5f47a 51c7d19b 36de3adf
        8833899d 7f27beb1 6a9152cf 765ee439
        0cce
```

16.1.2. SRTP AEAD_AES_128_GCM Decryption

Decrypting the following packet:

```
     8040f17b 8041f8d3 5501a0b2 f24de3a3
     fb34de6c acba861c 9d7e4bca be633bd5
     0d294e6f 42a5f47a 51c7d19b 36de3adf
     8833899d 7f27beb1 6a9152cf 765ee439
     0cce
```

Form the IV

```
       | Pad |   SSRC    |    ROC    | SEQ |
        00 00 55 01 a0 b2 00 00 00 00 f1 7b
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
```

```
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
AAD: 8040f17b 8041f8d3 5501a0b2
 CT: f24de3a3 fb34de6c acba861c 9d7e4bca
     be633bd5 0d294e6f 42a5f47a 51c7d19b
     36de3adf 8833899d 7f27beb1 6a9152cf
     765ee439 0cce
 IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
  H: c6a13b37878f5b826f4f8162a1c8d879
```

Verify the received tag

```
   89 9d 7f 27 be b1 6a 91 52 cf 76 5e e4 39 0c ce
```

Process the AAD

```
      AAD word: 8040f17b8041f8d35501a0b200000000
  partial hash: bcfb3d1d0e6e3e78ba45403377dba11b
```

Process the cipher

```
   cipher word: f24de3a3fb34de6cacba861c9d7e4bca
  partial hash: 0ebc0abe1b15b32fedd2b07888c1ef61
   cipher word: be633bd50d294e6f42a5f47a51c7d19b
  partial hash: 438e5797011ea860585709a2899f4685
   cipher word: 36de3adf883300000000000000000000
  partial hash: 336fb643310d7bac2aeaa76247f6036d
```

Process the length word

```
   length word: 00000000000000600000000000000130
  partial hash: 1b964067078c408c4e442a8f015e5264
```

```
   Turn GHASH into GMAC
              GHASH: 1b 96 40 67 07 8c 40 8c 4e 44 2a 8f 01 5e 52 64
                 K0: 92 0b 3f 40 b9 3d 2a 1d 1c 8b 5c d1 e5 67 5e aa
         full GMAC: 89 9d 7f 27 be b1 6a 91 52 cf 76 5e e4 39 0c ce

       Received tag = 899d7f27 beb16a91 52cf765e e4390cce
       Computed tag = 899d7f27 beb16a91 52cf765e e4390cce
     Received tag verified.

   Decrypt the cipher
     block # 0
       IV||blk_cntr: 51753c6580c2726f2071841400000002
          key_block: b5 2c 8f cf 92 55 fe 09 df ce a6 73 f0 10 22 b9
       cipher_block: f2 4d e3 a3 fb 34 de 6c ac ba 86 1c 9d 7e 4b ca
        plain_block: 47 61 6c 6c 69 61 20 65 73 74 20 6f 6d 6e 69 73
     block # 1
       IV||blk_cntr: 51753c6580c2726f2071841400000003
          key_block: 9e 07 52 a3 64 5a 2f 4f 2b cb d4 0a 30 b5 a5 fe
       cipher_block: be 63 3b d5 0d 29 4e 6f 42 a5 f4 7a 51 c7 d1 9b
        plain_block: 20 64 69 76 69 73 61 20 69 6e 20 70 61 72 74 65
     block # 2
       IV||blk_cntr: 51753c6580c2726f2071841400000004
          key_block: 45 fe 4e ad ed 40 0a 5d 1a f3 63 f9 0c e1 49 3b
       cipher_block: 36 de 3a df 88 33
        plain_block: 73 20 74 72 65 73

   Verified and tagged packet:
        47616c6c 69612065 7374206f 6d6e6973
        20646976 69736120 696e2070 61727465
        73207472 6573
```

16.1.3. SRTP AEAD_AES_128_GCM Authentication Tagging

Tagging the following packet:

```
     8040f17b 8041f8d3 5501a0b2 47616c6c
     69612065 7374206f 6d6e6973 20646976
     69736120 696e2070 61727465 73207472
     6573
```

Form the IV

```
       | Pad |   SSRC    |    ROC    | SEQ |
         00 00 55 01 a0 b2 00 00 00 00 f1 7b
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
```

```
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
AAD: 8040f17b 8041f8d3 5501a0b2 47616c6c
     69612065 7374206f 6d6e6973 20646976
     69736120 696e2070 61727465 73207472
     6573
 IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
  H: c6a13b37878f5b826f4f8162a1c8d879
```

Compute the GMAC tag

```
  Process the AAD
        AAD word: 8040f17b8041f8d35501a0b247616c6c
    partial hash: 79f41fea34a474a77609d8925e9f2b22
        AAD word: 696120657374206f6d6e697320646976
    partial hash: 84093a2f85abf17ab37d3ce2f706138f
        AAD word: 69736120696e20706172746573207472
    partial hash: ab2760fee24e6dec754739d8059cd144
        AAD word: 65730000000000000000000000000000
    partial hash: e84f3c55d287fc561c41d09a8aada4be

  Process the length word
     length word: 00000000000001900000000000000000
    partial hash: b04200c26b81c98af55cc2eafccd1cbc
```

```
Turn GHASH into GMAC
           GHASH: b0 42 00 c2 6b 81 c9 8a f5 5c c2 ea fc cd 1c bc
              K0: 92 0b 3f 40 b9 3d 2a 1d 1c 8b 5c d1 e5 67 5e aa
       full GMAC: 22 49 3f 82 d2 bc e3 97 e9 d7 9e 3b 19 aa 42 16
```

Cipher with tag

```
     22493f82 d2bce397 e9d79e3b 19aa4216
```

```
   Tagged packet:
        8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        65732249 3f82d2bc e397e9d7 9e3b19aa
        4216
```

16.1.4. SRTP AEAD_AES_128_GCM Tag Verification

```
   Verifying the following packet:

        8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        65732249 3f82d2bc e397e9d7 9e3b19aa
        4216

   Form the IV
          | Pad |   SSRC    |     ROC     | SEQ |
           00 00 55 01 a0 b2 00 00 00 00 f1 7b
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
   AAD: 8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        6573
    CT: 22493f82 d2bce397 e9d79e3b 19aa4216
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
     H: c6a13b37878f5b826f4f8162a1c8d879

   Verify the received tag
      22 49 3f 82 d2 bc e3 97 e9 d7 9e 3b 19 aa 42 16

     Process the AAD
           AAD word: 8040f17b8041f8d35501a0b247616c6c
       partial hash: 79f41fea34a474a77609d8925e9f2b22
           AAD word: 696120657374206f6d6e697320646976
       partial hash: 84093a2f85abf17ab37d3ce2f706138f
           AAD word: 69736120696e20706172746573207472
       partial hash: ab2760fee24e6dec754739d8059cd144
           AAD word: 65730000000000000000000000000000
       partial hash: e84f3c55d287fc561c41d09a8aada4be

     Process the length word
          length word: 00000000000001900000000000000000
       partial hash: b04200c26b81c98af55cc2eafccd1cbc
```

```
   Turn GHASH into GMAC
              GHASH: b0 42 00 c2 6b 81 c9 8a f5 5c c2 ea fc cd 1c bc
                 K0: 92 0b 3f 40 b9 3d 2a 1d 1c 8b 5c d1 e5 67 5e aa
           full GMAC: 22 49 3f 82 d2 bc e3 97 e9 d7 9e 3b 19 aa 42 16

        Received tag = 22493f82 d2bce397 e9d79e3b 19aa4216
        Computed tag = 22493f82 d2bce397 e9d79e3b 19aa4216
     Received tag verified.
```

16.2. SRTP AEAD_AES_256_GCM

16.2.1. SRTP AEAD_AES_256_GCM Encryption

```
   Encrypting the following packet:

        8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        6573

   Form the IV
          | Pad |   SSRC    |    ROC    | SEQ |
           00 00 55 01 a0 b2 00 00 00 00 f1 7b
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
        10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
   AAD: 8040f17b 8041f8d3 5501a0b2
    PT: 47616c6c 69612065 7374206f 6d6e6973
        20646976 69736120 696e2070 61727465
        73207472 6573
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
     H: f29000b62a499fd0a9f39a6add2e7780
```

```
   Encrypt the Plaintext
     block # 0
       IV||blk_cntr: 51753c6580c2726f2071841400000002
          key_block: 75 d0 b2 14 c1 43 de 77 9c eb 58 95 5e 40 5a d9
        plain_block: 47 61 6c 6c 69 61 20 65 73 74 20 6f 6d 6e 69 73
       cipher_block: 32 b1 de 78 a8 22 fe 12 ef 9f 78 fa 33 2e 33 aa
     block # 1
       IV||blk_cntr: 51753c6580c2726f2071841400000003
          key_block: 91 e4 7b 4e f3 2b 83 d3 dc 65 0a 72 17 8d da 6a
        plain_block: 20 64 69 76 69 73 61 20 69 6e 20 70 61 72 74 65
       cipher_block: b1 80 12 38 9a 58 e2 f3 b5 0b 2a 02 76 ff ae 0f
     block # 2
       IV||blk_cntr: 51753c6580c2726f2071841400000004
          key_block: 68 86 43 eb dd 08 07 98 16 3a 16 d5 e5 04 f6 3a
        plain_block: 73 20 74 72 65 73
       cipher_block: 1b a6 37 99 b8 7b

   Cipher before tag appended
        32b1de78 a822fe12 ef9f78fa 332e33aa
        b1801238 9a58e2f3 b50b2a02 76ffae0f
        1ba63799 b87b

   Compute the GMAC tag

     Process the AAD
           AAD word: 8040f17b8041f8d35501a0b200000000
       partial hash: 0154dcb75485b71880e1957c877351bd

     Process the cipher
        cipher word: 32b1de78a822fe12ef9f78fa332e33aa
       partial hash: c3f07db9a8b9cb4345eb07f793d322d2
        cipher word: b18012389a58e2f3b50b2a0276ffae0f
       partial hash: 6d1e66fe32eb32ecd8906ceab09db996
        cipher word: 1ba63799b87b0000000000000000000
       partial hash: b3d1d2f1fa3b366619bc42cd2eedafee

     Process the length word
        length word: 00000000000000600000000000000130
       partial hash: 7debf5fa1fac3bd318d5e1a7ee401091

   Turn GHASH into GMAC
               GHASH: 7d eb f5 fa 1f ac 3b d3 18 d5 e1 a7 ee 40 10 91
                  K0: 07 48 2e cc c0 53 ed 63 e1 6e 99 df 39 e7 7c 82
          full GMAC: 7a a3 db 36 df ff d6 b0 f9 bb 78 78 d7 a7 6c 13
```

```
   Cipher with tag
        32b1de78 a822fe12 ef9f78fa 332e33aa
        b1801238 9a58e2f3 b50b2a02 76ffae0f
        1ba63799 b87b7aa3 db36dfff d6b0f9bb
        7878d7a7 6c13

   Encrypted and tagged packet:
        8040f17b 8041f8d3 5501a0b2 32b1de78
        a822fe12 ef9f78fa 332e33aa b1801238
        9a58e2f3 b50b2a02 76ffae0f 1ba63799
        b87b7aa3 db36dfff d6b0f9bb 7878d7a7
        6c13
```

16.2.2. SRTP AEAD_AES_256_GCM Decryption

```
   Decrypting the following packet:

        8040f17b 8041f8d3 5501a0b2 32b1de78
        a822fe12 ef9f78fa 332e33aa b1801238
        9a58e2f3 b50b2a02 76ffae0f 1ba63799
        b87b7aa3 db36dfff d6b0f9bb 7878d7a7
        6c13

   Form the IV
          | Pad |   SSRC    |    ROC    | SEQ |
           00 00 55 01 a0 b2 00 00 00 00 f1 7b
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
        10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
   AAD: 8040f17b 8041f8d3 5501a0b2
    CT: 32b1de78 a822fe12 ef9f78fa 332e33aa
        b1801238 9a58e2f3 b50b2a02 76ffae0f
        1ba63799 b87b7aa3 db36dfff d6b0f9bb
        7878d7a7 6c13
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
     H: f29000b62a499fd0a9f39a6add2e7780

   Verify the received tag
      7a a3 db 36 df ff d6 b0 f9 bb 78 78 d7 a7 6c 13

     Process the AAD
           AAD word: 8040f17b8041f8d35501a0b200000000
       partial hash: 0154dcb75485b71880e1957c877351bd
```

```
     Process the cipher
        cipher word: 32b1de78a822fe12ef9f78fa332e33aa
      partial hash: c3f07db9a8b9cb4345eb07f793d322d2
        cipher word: b18012389a58e2f3b50b2a0276ffae0f
      partial hash: 6d1e66fe32eb32ecd8906ceab09db996
        cipher word: 1ba63799b87b00000000000000000000
      partial hash: b3d1d2f1fa3b366619bc42cd2eedafee

     Process the length word
        length word: 00000000000000060000000000000130
      partial hash: 7debf5fa1fac3bd318d5e1a7ee401091

   Turn GHASH into GMAC
              GHASH: 7d eb f5 fa 1f ac 3b d3 18 d5 e1 a7 ee 40 10 91
                 K0: 07 48 2e cc c0 53 ed 63 e1 6e 99 df 39 e7 7c 82
         full GMAC: 7a a3 db 36 df ff d6 b0 f9 bb 78 78 d7 a7 6c 13

        Received tag = 7aa3db36 dfffd6b0 f9bb7878 d7a76c13
        Computed tag = 7aa3db36 dfffd6b0 f9bb7878 d7a76c13
     Received tag verified.

   Decrypt the cipher
     block # 0
       IV||blk_cntr: 51753c6580c2726f2071841400000002
          key_block: 75 d0 b2 14 c1 43 de 77 9c eb 58 95 5e 40 5a d9
       cipher_block: 32 b1 de 78 a8 22 fe 12 ef 9f 78 fa 33 2e 33 aa
        plain_block: 47 61 6c 6c 69 61 20 65 73 74 20 6f 6d 6e 69 73
     block # 1
       IV||blk_cntr: 51753c6580c2726f2071841400000003
          key_block: 91 e4 7b 4e f3 2b 83 d3 dc 65 0a 72 17 8d da 6a
       cipher_block: b1 80 12 38 9a 58 e2 f3 b5 0b 2a 02 76 ff ae 0f
        plain_block: 20 64 69 76 69 73 61 20 69 6e 20 70 61 72 74 65
     block # 2
       IV||blk_cntr: 51753c6580c2726f2071841400000004
          key_block: 68 86 43 eb dd 08 07 98 16 3a 16 d5 e5 04 f6 3a
       cipher_block: 1b a6 37 99 b8 7b
        plain_block: 73 20 74 72 65 73

   Verified and tagged packet:
        47616c6c 69612065 7374206f 6d6e6973
        20646976 69736120 696e2070 61727465
        73207472 6573
```

16.2.3. SRTP AEAD_AES_256_GCM Authentication Tagging

Tagging the following packet:

```
     8040f17b 8041f8d3 5501a0b2 47616c6c
     69612065 7374206f 6d6e6973 20646976
     69736120 696e2070 61727465 73207472
     6573
```

Form the IV

```
       | Pad |   SSRC    |    ROC    | SEQ |
         00 00 55 01 a0 b2 00 00 00 00 f1 7b
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
```

```
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
     10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
AAD: 8040f17b 8041f8d3 5501a0b2 47616c6c
     69612065 7374206f 6d6e6973 20646976
     69736120 696e2070 61727465 73207472
     6573
 IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
  H: f29000b62a499fd0a9f39a6add2e7780
```

Compute the GMAC tag

```
  Process the AAD
       AAD word: 8040f17b8041f8d35501a0b247616c6c
    partial hash: c059753e6763791762ca630d8ef97714
       AAD word: 696120657374206f6d6e697320646976
    partial hash: a4e3401e712900dc4f1d2303bc4b2675
       AAD word: 69736120696e20706172746573207472
    partial hash: 1c8c1af883de0d67878f379a19c65987
       AAD word: 65730000000000000000000000000000
    partial hash: 958462781aa8e8feacce6d93b54472ac

  Process the length word
     length word: 00000000000001900000000000000000
    partial hash: af2efb5dcfdb9900e7127721fdb56956
```

Turn GHASH into GMAC

```
          GHASH: af 2e fb 5d cf db 99 00 e7 12 77 21 fd b5 69 56
             K0: 07 48 2e cc c0 53 ed 63 e1 6e 99 df 39 e7 7c 82
      full GMAC: a8 66 d5 91 0f 88 74 63 06 7c ee fe c4 52 15 d4
```

Cipher with tag

```
     a866d591 0f887463 067ceefe c45215d4
```

```
   Tagged packet:
        8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        6573a866 d5910f88 7463067c eefec452
        15d4
```

16.2.4. SRTP AEAD_AES_256_GCM Tag Verification

```
   Verifying the following packet:

        8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        6573a866 d5910f88 7463067c eefec452
        15d4

   Form the IV
          | Pad |   SSRC    |     ROC     | SEQ |
           00 00 55 01 a0 b2 00 00 00 00 f1 7b
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
        10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
   AAD: 8040f17b 8041f8d3 5501a0b2 47616c6c
        69612065 7374206f 6d6e6973 20646976
        69736120 696e2070 61727465 73207472
        6573
    CT: a866d591 0f887463 067ceefe c45215d4
    IV: 51 75 3c 65 80 c2 72 6f 20 71 84 14
     H: f29000b62a499fd0a9f39a6add2e7780

   Verify the received tag
      a8 66 d5 91 0f 88 74 63 06 7c ee fe c4 52 15 d4

     Process the AAD
           AAD word: 8040f17b8041f8d35501a0b247616c6c
       partial hash: c059753e6763791762ca630d8ef97714
           AAD word: 696120657374206f6d6e697320646976
       partial hash: a4e3401e712900dc4f1d2303bc4b2675
           AAD word: 69736120696e20706172746573207472
       partial hash: 1c8c1af883de0d67878f379a19c65987
           AAD word: 65730000000000000000000000000000
       partial hash: 958462781aa8e8feacce6d93b54472ac
```

```
     Process the length word
        length word: 00000000000001900000000000000000
      partial hash: af2efb5dcfdb9900e7127721fdb56956

   Turn GHASH into GMAC
              GHASH: af 2e fb 5d cf db 99 00 e7 12 77 21 fd b5 69 56
                 K0: 07 48 2e cc c0 53 ed 63 e1 6e 99 df 39 e7 7c 82
         full GMAC: a8 66 d5 91 0f 88 74 63 06 7c ee fe c4 52 15 d4

        Received tag = a866d591 0f887463 067ceefe c45215d4
        Computed tag = a866d591 0f887463 067ceefe c45215d4
     Received tag verified.
```

17. RTCP Test Vectors

The examples in this section are all based upon the same RTCP packet:

```
          81c8000e 4d617273 4e545031 4e545031
          52545020 0000042a 0000eb98 4c756e61
          deadbeef deadbeef deadbeef deadbeef
          deadbeef
```

with 32-bit SRTCP index 000005d4.

As shown in Section 9.1, the IV is formed by XORing two 12-octet values. The first 12-octet value is formed by concatenating two zero octets, the 4-octet SSRC (found in the fifth through eighth octets of the RTP packet), another two padding octets, and the 31-bit SRTCP index, right-justified in a 32-bit = 4-octet field with a single "0" bit prepended as padding. An example of SRTCP IV formation is shown below:

```
           | Pad |   SSRC    | Pad |  0+SRTCP  |
            00 00 4d 61 72 73 00 00 00 00 05 d4
    salt    51 75 69 64 20 70 72 6f 20 71 75 6f
            -----------------------------------
      IV    51 75 24 05 52 03 72 6f 20 71 70 bb
```

In an SRTCP packet, a 1-bit Encryption flag is prepended to the 31-bit SRTCP index to form a 32-bit value we shall call the "ESRTCP word". The E-flag is one if the SRTCP packet has been encrypted and zero if it has been tagged but not encrypted. Note that the ESRTCP field is only present in an SRTCP packet, not in an RTCP packet. The full ESRTCP word is part of the AAD.

When encrypting and tagging an RTCP packet (E-flag = 1), the SRTCP packet consists of the following fields in the following order:

- The first 8 octets of the RTCP packet (part of the AAD).

- The cipher.

- The ESRTCP word (the final part of the AAD).

- Any Raw Data that might have been appended to the end of the original RTCP packet.

Recall that AEAD treats the authentication tag as an integral part of the cipher, and in fact the authentication tag is the last 8 or 16 octets of the cipher.

The reader is reminded that when the RTCP packet is to be tagged but not encrypted (E-flag = 0), GCM will produce a cipher that consists solely of the 8-octet or 16-octet authentication tag. The tagged SRTCP consists of the following fields in the order listed below:

- All of the AAD, except for the ESRTCP word.

- The cipher (= the authentication tag).

- The ESRTCP word (the final part of the AAD).

- Any Raw Data that might have been appended to the end of the original RTCP packet.

17.1. SRTCP AEAD_AES_128_GCM Encryption and Tagging

Encrypting the following packet:

```
     81c8000d 4d617273 4e545031 4e545032
     52545020 0000042a 0000e930 4c756e61
     deadbeef deadbeef deadbeef deadbeef
     deadbeef
```

```
Key size = 128 bits
Tag size =  16 octets

Form the IV
       | Pad |   SSRC    | Pad |   SRTCP   |
        00 00 4d 61 72 73 00 00 00 00 05 d4
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 24 05 52 03 72 6f 20 71 70 bb

Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
AAD: 81c8000d 4d617273 800005d4
 PT: 4e545031 4e545032 52545020 0000042a
     0000e930 4c756e61 deadbeef deadbeef
     deadbeef deadbeef deadbeef
 IV: 51 75 24 05 52 03 72 6f 20 71 70 bb
  H: c6a13b37878f5b826f4f8162a1c8d879

Encrypt the Plaintext
  block # 0
    IV||blk_cntr: 517524055203726f207170bb00000002
       key_block: 2d bd 18 b4 92 8e e6 4e f5 73 87 46 2f 6b 7a b3
     plain_block: 4e 54 50 31 4e 54 50 32 52 54 50 20 00 00 04 2a
    cipher_block: 63 e9 48 85 dc da b6 7c a7 27 d7 66 2f 6b 7e 99
  block # 1
    IV||blk_cntr: 517524055203726f207170bb00000003
       key_block: 7f f5 29 c7 20 73 9d 4c 18 db 1b 1e ad a0 d1 35
     plain_block: 00 00 e9 30 4c 75 6e 61 de ad be ef de ad be ef
    cipher_block: 7f f5 c0 f7 6c 06 f3 2d c6 76 a5 f1 73 0d 6f da
  block # 2
    IV||blk_cntr: 517524055203726f207170bb00000004
       key_block: 92 4d 25 a9 58 9d 83 02 d5 14 99 b4 e0 14 78 15
     plain_block: de ad be ef de ad be ef de ad be ef
    cipher_block: 4c e0 9b 46 86 30 3d ed 0b b9 27 5b

Cipher before tag appended
     63e94885 dcdab67c a727d766 2f6b7e99
     7ff5c0f7 6c06f32d c676a5f1 730d6fda
     4ce09b46 86303ded 0bb9275b
```

```
   Compute the GMAC tag

     Process the AAD
           AAD word: 81c8000d4d617273800005d400000000
       partial hash: 085d6eb166c555aa62982f630430ec6e

     Process the cipher
        cipher word: 63e94885dcdab67ca727d7662f6b7e99
       partial hash: 8c9221be93466d68bbb16fa0d42b0187
        cipher word: 7ff5c0f76c06f32dc676a5f1730d6fda
       partial hash: 221ebb044ec9fd0bf116d7780f198792
        cipher word: 4ce09b4686303ded0bb9275b00000000
       partial hash: 50f70b9ca110ab312dce212657328dae

     Process the length word
        length word: 00000000000000060000000000000160
       partial hash: 7296107c9716534371dfc1a30c5ffeb5

   Turn GHASH into GMAC
              GHASH: 72 96 10 7c 97 16 53 43 71 df c1 a3 0c 5f fe b5
                 K0: ba dc b4 24 01 d9 1e 6c b4 74 39 d1 49 86 14 6b
          full GMAC: c8 4a a4 58 96 cf 4d 2f c5 ab f8 72 45 d9 ea de

   Cipher with tag
        63e94885 dcdab67c a727d766 2f6b7e99
        7ff5c0f7 6c06f32d c676a5f1 730d6fda
        4ce09b46 86303ded 0bb9275b c84aa458
        96cf4d2f c5abf872 45d9eade

   Append the ESRTCP word with the E-flag set
        63e94885 dcdab67c a727d766 2f6b7e99
        7ff5c0f7 6c06f32d c676a5f1 730d6fda
        4ce09b46 86303ded 0bb9275b c84aa458
        96cf4d2f c5abf872 45d9eade 800005d4

   Encrypted and tagged packet:
        81c8000d 4d617273 63e94885 dcdab67c
        a727d766 2f6b7e99 7ff5c0f7 6c06f32d
        c676a5f1 730d6fda 4ce09b46 86303ded
        0bb9275b c84aa458 96cf4d2f c5abf872
        45d9eade 800005d4
```

17.2. SRTCP AEAD_AES_256_GCM Verification and Decryption

```
   Key size = 256 bits
   Tag size =  16 octets

     Process the length word

   Decrypting the following packet:

        81c8000d 4d617273 d50ae4d1 f5ce5d30
        4ba297e4 7d470c28 2c3ece5d bffe0a50
        a2eaa5c1 110555be 8415f658 c61de047
        6f1b6fad 1d1eb30c 4446839f 57ff6f6c
        b26ac3be 800005d4

   Key size = 256 bits
   Key size =  16 octets

   Form the IV
          | Pad |   SSRC    | Pad |   SRTCP   |
           00 00 4d 61 72 73 00 00 00 00 05 d4
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 24 05 52 03 72 6f 20 71 70 bb

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
        10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
   AAD: 81c8000d 4d617273 800005d4
    CT: d50ae4d1 f5ce5d30 4ba297e4 7d470c28
        2c3ece5d bffe0a50 a2eaa5c1 110555be
        8415f658 c61de047 6f1b6fad 1d1eb30c
        4446839f 57ff6f6c b26ac3be
    IV: 51 75 24 05 52 03 72 6f 20 71 70 bb
     H: f29000b62a499fd0a9f39a6add2e7780

   Verify the received tag
      1d 1e b3 0c 44 46 83 9f 57 ff 6f 6c b2 6a c3 be

     Process the AAD
           AAD word: 81c8000d4d617273800005d400000000
       partial hash: 3ae5afd36dead5280b18950400176b5b

     Process the cipher
        cipher word: d50ae4d1f5ce5d304ba297e47d470c28
       partial hash: e90fab7546f6940781227227ac926ebe
        cipher word: 2c3ece5dbffe0a50a2eaa5c1110555be
       partial hash: 9b236807d8b2dab07583adce367aa88f
        cipher word: 8415f658c61de0476f1b6fad00000000
       partial hash: e69313f423a75e3e0b7eb93321700e86
```

```
    Process the length word
       length word: 00000000000000600000000000000160
      partial hash: 3a284af2616fdf505faf37eec39fbc8b

  Turn GHASH into GMAC
             GHASH: 3a 28 4a f2 61 6f df 50 5f af 37 ee c3 9f bc 8b
                K0: 27 36 f9 fe 25 29 5c cf 08 50 58 82 71 f5 7f 35
        full GMAC: 1d 1e b3 0c 44 46 83 9f 57 ff 6f 6c b2 6a c3 be

       Received tag = 1d1eb30c 4446839f 57ff6f6c b26ac3be
       Computed tag = 1d1eb30c 4446839f 57ff6f6c b26ac3be
    Received tag verified.

  Decrypt the cipher
    block # 0
      IV||blk_cntr: 517524055203726f207170bb00000002
         key_block: 9b 5e b4 e0 bb 9a 0d 02 19 f6 c7 c4 7d 47 08 02
      cipher_block: d5 0a e4 d1 f5 ce 5d 30 4b a2 97 e4 7d 47 0c 28
       plain_block: 4e 54 50 31 4e 54 50 32 52 54 50 20 00 00 04 2a
    block # 1
      IV||blk_cntr: 517524055203726f207170bb00000003
         key_block: 2c 3e 27 6d f3 8b 64 31 7c 47 1b 2e cf a8 eb 51
      cipher_block: 2c 3e ce 5d bf fe 0a 50 a2 ea a5 c1 11 05 55 be
       plain_block: 00 00 e9 30 4c 75 6e 61 de ad be ef de ad be ef
    block # 2
      IV||blk_cntr: 517524055203726f207170bb00000004
         key_block: 5a b8 48 b7 18 b0 5e a8 b1 b6 d1 42 3b 74 39 55
      cipher_block: 84 15 f6 58 c6 1d e0 47 6f 1b 6f ad
       plain_block: de ad be ef de ad be ef de ad be ef

  Verified and decrypted packet:
       81c8000d 4d617273 4e545031 4e545032
       52545020 0000042a 0000e930 4c756e61
       deadbeef deadbeef deadbeef deadbeef
       deadbeef
```

17.3. SRTCP AEAD_AES_128_GCM Tagging Only

Tagging the following packet:

```
     81c8000d 4d617273 4e545031 4e545032
     52545020 0000042a 0000e930 4c756e61
     deadbeef deadbeef deadbeef deadbeef
     deadbeef
```

```
Key size = 128 bits
Tag size =  16 octets
```

Form the IV

```
       | Pad |   SSRC    | Pad |   SRTCP   |
        00 00 4d 61 72 73 00 00 00 00 05 d4
  salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
    IV: 51 75 24 05 52 03 72 6f 20 71 70 bb
```

```
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
AAD: 81c8000d 4d617273 4e545031 4e545032
     52545020 0000042a 0000e930 4c756e61
     deadbeef deadbeef deadbeef deadbeef
     deadbeef 000005d4
 IV: 51 75 24 05 52 03 72 6f 20 71 70 bb
  H: c6a13b37878f5b826f4f8162a1c8d879
```

Compute the GMAC tag

Process the AAD

```
      AAD word: 81c8000d4d6172734e5450314e545032
partial hash: f8dbbe278e06afe17fb4fb2e67f0a22e
      AAD word: 525450200000042a0000e9304c756e61
partial hash: 6ccd900dfd0eb292f68f8a410d0648ec
      AAD word: deadbeefdeadbeefdeadbeefdeadbeef
partial hash: 6a14be0ea384c6b746235ba955a57ff5
      AAD word: deadbeef000005d40000000000000000
partial hash: cc81f14905670a1e37f8bc81a91997cd
```

Process the length word

```
   length word: 00000000000001c0000000000000000
partial hash: 3ec16d4c3c0e90a59e91be415bd976d8
```

Turn GHASH into GMAC

```
          GHASH: 3e c1 6d 4c 3c 0e 90 a5 9e 91 be 41 5b d9 76 d8
             K0: ba dc b4 24 01 d9 1e 6c b4 74 39 d1 49 86 14 6b
      full GMAC: 84 1d d9 68 3d d7 8e c9 2a e5 87 90 12 5f 62 b3
```

```
   Cipher with tag
        841dd968 3dd78ec9 2ae58790 125f62b3

   Tagged packet:
        81c8000d 4d617273 4e545031 4e545032
        52545020 0000042a 0000e930 4c756e61
        deadbeef deadbeef deadbeef deadbeef
        deadbeef 841dd968 3dd78ec9 2ae58790
        125f62b3 000005d4
```

17.4. SRTCP AEAD_AES_256_GCM Tag Verification

```
   Key size = 256 bits
   Tag size =  16 octets

     Process the length word
   Verifying the following packet:

        81c8000d 4d617273 4e545031 4e545032
        52545020 0000042a 0000e930 4c756e61
        deadbeef deadbeef deadbeef deadbeef
        deadbeef 91db4afb feee5a97 8fab4393
        ed2615fe 000005d4

   Key size = 256 bits
   Key size =  16 octets

   Form the IV
          | Pad |   SSRC    | Pad |   SRTCP   |
           00 00 4d 61 72 73 00 00 00 00 05 d4
     salt: 51 75 69 64 20 70 72 6f 20 71 75 6f
       IV: 51 75 24 05 52 03 72 6f 20 71 70 bb

   Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
        10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
   AAD: 81c8000d 4d617273 4e545031 4e545032
        52545020 0000042a 0000e930 4c756e61
        deadbeef deadbeef deadbeef deadbeef
        deadbeef 000005d4
    CT: 91db4afb feee5a97 8fab4393 ed2615fe
    IV: 51 75 24 05 52 03 72 6f 20 71 70 bb
     H: f29000b62a499fd0a9f39a6add2e7780

   Verify the received tag
      91 db 4a fb fe ee 5a 97 8f ab 43 93 ed 26 15 fe
```

```
     Process the AAD
           AAD word: 81c8000d4d6172734e5450314e545032
       partial hash: 7bc665c71676a5a5f663b3229af4b85c
           AAD word: 525450200000042a0000e9304c756e61
       partial hash: 34ed77752703ab7d69f44237910e3bc0
           AAD word: deadbeefdeadbeefdeadbeefdeadbeef
       partial hash: 74a59f1a99282344d64ab1c8a2be6cf8
           AAD word: deadbeef000005d40000000000000000
       partial hash: 126335c0baa7ab1b79416ceeb9f7a518

     Process the length word
        length word: 00000000000001c00000000000000000
       partial hash: b6edb305dbc7065887fb1b119cd36acb

   Turn GHASH into GMAC
              GHASH: b6 ed b3 05 db c7 06 58 87 fb 1b 11 9c d3 6a cb
                 K0: 27 36 f9 fe 25 29 5c cf 08 50 58 82 71 f5 7f 35
          full GMAC: 91 db 4a fb fe ee 5a 97 8f ab 43 93 ed 26 15 fe

        Received tag = 91db4afb feee5a97 8fab4393 ed2615fe
        Computed tag = 91db4afb feee5a97 8fab4393 ed2615fe
     Received tag verified.

   Verified packet:
        81c8000d 4d617273 4e545031 4e545032
        52545020 0000042a 0000e930 4c756e61
        deadbeef deadbeef deadbeef deadbeef
        deadbeef
```

18. References

18.1. 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>.

[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>.

[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>.

[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>.

[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>.

[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>.

18.2. Informative References

[BN00] Bellare, M. and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", Proceedings of ASIACRYPT 2000, Springer-Verlag, LNCS 1976, pp. 531-545, DOI 10.1007/3-540-44448-3_41, <http://www-cse.ucsd.edu/users/mihir/papers/oem.html>.

[GCM] Dworkin, M., "NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", U.S. National Institute of Standards and Technology, November 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

[R02] Rogaway, P., "Authenticated-Encryption with Associated-Data", ACM Conference on Computer and Communications Security (CCS'02), pp. 98-107, ACM Press, DOI 10.1145/586110.586125, September 2002, <http://www.cs.ucdavis.edu/~rogaway/papers/ad.html>.

[RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, DOI 10.17487/RFC4771, January 2007, <http://www.rfc-editor.org/info/rfc4771>.

Acknowledgements

The authors would like to thank Michael Peck, Michael Torla, Qin Wu, Magnus Westerlund, Oscar Ohllson, Woo-Hwan Kim, John Mattsson, Richard Barnes, Morris Dworkin, Stephen Farrell, and many other reviewers who provided valuable comments on earlier draft versions of this document.

Authors' Addresses

David A. McGrew
Cisco Systems, Inc.
510 McCarthy Blvd.
Milpitas, CA 95035
United States
Phone: (408) 525 8651

Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm

Kevin M. Igoe
NSA/CSS Commercial Solutions Center
National Security Agency

Email: mythicalkevin@yahoo.com