Internet Engineering Task Force
AVT Working Group
INTERNET-DRAFT
EXPIRES: December 2003

Baugher, McGrew (Cisco)
Carrara, Naslund,
Norrman (Ericsson)

July 2003

The Secure Real-time Transport Protocol
<draft-ietf-avt-srtp-09.txt>

Status of this memo

Abstract

This document describes the Secure Real-time Transport Protocol (SRTP), a profile of the Real-time Transport Protocol (RTP), 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).

TABLE OF CONTENTS

1. Introduction

This document describes the Secure Real-time Transport Protocol (SRTP), a profile of the Real-time Transport Protocol (RTP), which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the control traffic for RTP, RTCP (the Real-time Transport Control Protocol) [RTPNEW].

SRTP provides a framework for encryption and message authentication of RTP and RTCP streams (Section 3). SRTP defines a set of default cryptographic transforms (Sections 4 and 5), and it allows new transforms to be introduced in the future (Section 6). With appropriate key management (Sections 7 and 8), SRTP is secure (Sections 9) for unicast and multicast RTP applications (Section 11).

SRTP can achieve high throughput and low packet expansion. SRTP proves to be a suitable protection for heterogeneous environments (mix of wired and wireless networks). To get such features, default transforms are described, based on an additive stream cipher for encryption, a keyed-hash based function for message authentication, and an "implicit" index for sequencing/synchronization based on the RTP sequence number for SRTP and an index number for Secure RTCP (SRTCP).

1.1. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
The terminology conforms to [RFC2828] with the following exception. For simplicity we use the term "random" throughout the document to denote randomly or pseudo-randomly generated values. Large amounts of random bits may be difficult to obtain, and for the security of SRTP, pseudo-randomness is sufficient.

By convention, the adopted representation is the network byte order, i.e. the left most bit (octet) is the most significant one. By XOR we mean bitwise addition modulo 2 of binary strings, and || denotes concatenation. In other words, if C = A || B, then the most significant bits of C are the bits of A, and the least significant bits of C equal the bits of B. Hexadecimal numbers are prefixed by 0x.

The word "encryption" includes also use of the NULL algorithm (which in practice does leave the data in the clear).

With slight abuse of notation, we use the terms "message authentication" and "authentication tag" as is common practice, even though in some circumstances, e.g. group communication, the service provided is actually only integrity protection and not data origin authentication.

2. Goals and Features

The security goals for SRTP are to ensure:

* the confidentiality of the RTP and RTCP payloads, and

* the integrity of the entire RTP and RTCP packets, together with protection against replayed packets.

These security services are optional and independent from each other, except that SRTCP integrity protection is mandatory (malicious or erroneous alteration of RTCP messages could otherwise disrupt the processing of the RTP stream).

Other, functional, goals for the protocol are:

* a framework that permits upgrading with new cryptographic transforms,

* low bandwidth cost, i.e., a framework preserving RTP header compression efficiency,

and, asserted by the pre-defined transforms:

* a low computational cost,

* a small footprint (i.e. small code size and data memory for keying information and replay lists),

* limited packet expansion to support the bandwidth economy goal,

* independence from the underlying transport, network, and physical layers used by RTP, in particular high tolerance to packet loss and re-ordering.

These properties ensure that SRTP is a suitable protection scheme for RTP/RTCP in both wired and wireless scenarios.

2.1 Features

Besides the above mentioned direct goals, SRTP provides for some additional features. They have been introduced to lighten the burden on key management and to further increase security. They include:

* A single "master key" can provide keying material for confidentiality and integrity protection, both for the SRTP stream and the corresponding SRTCP stream. This is achieved with a key derivation function (see Section 4.3), providing "session keys" for the respective security primitive, securely derived from the master key.

* In addition, the key derivation can be configured to periodically refresh the session keys, which limits the amount of ciphertext produced by a fixed key, available for an adversary to cryptanalyze.

* "Salting keys" are used to protect against pre-computation and time-memory tradeoff attacks [MF00,BS00].

Detailed rationale for these features can be found in Section 7.

3. SRTP Framework

RTP is the Real-time Transport Protocol [RTPNEW]. We define SRTP as a profile of RTP. This profile is an extension to the RTP Audio/Video Profile [AVPNEW]. Except where explicitly noted, all

aspects of that profile apply, with the addition of the SRTP security features. Conceptually, we consider SRTP to be a "bump in the stack" implementation which resides between the RTP application and the transport layer. SRTP intercepts RTP packets and then forwards an equivalent SRTP packet on the sending side, and intercepts SRTP packets and passes an equivalent RTP packet up the stack on the receiving side.

Secure RTCP (SRTCP) provides the same security services to RTCP as SRTP does to RTP. SRTCP message authentication is MANDATORY and thereby protects the RTCP fields to keep track of membership, provide feedback to RTP senders, or maintain packet sequence counters. SRTCP is described in Section 3.4.

3.1 Secure RTP

The format of an SRTP packet is illustrated in Figure 1.

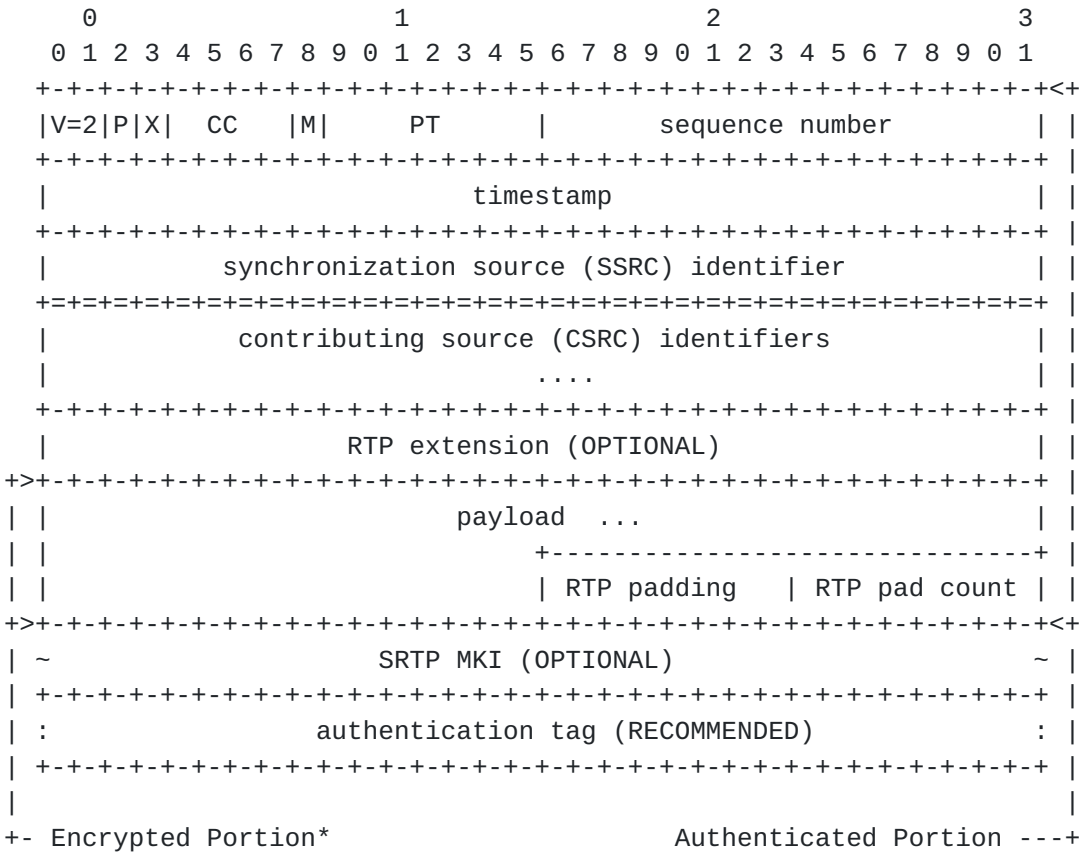

Figure 1. The format of an SRTP packet. *Encrypted Portion is the same size as the plaintext for the Section 4 pre-defined transforms.

The "Encrypted Portion" of an SRTP packet consists of the encryption of the RTP payload (including RTP padding when present) of the equivalent RTP packet. The Encrypted Portion MAY be the exact size of the plaintext or MAY be larger. Figure 1 shows the RTP payload including any possible padding for RTP [RTPNEW].

None of the pre-defined encryption transforms uses any padding; for these, the RTP and SRTP payload sizes match exactly. New transforms added to SRTP (following Section 6) may require padding, and may hence produce larger payloads. RTP provides its own padding format (as seen in Fig. 1), which due to the padding indicator in the RTP header has merits in terms of compactness relative to paddings using prefix-free codes. This RTP padding SHALL be the default method for transforms requiring padding. Transforms MAY specify other padding methods, and MUST then specify the amount, format, and processing of their padding. It is important to note that encryption transforms that use padding are vulnerable to subtle attacks, especially when message authentication is not used [V02]. Each specification for a new encryption transform needs to carefully consider and describe the security implications of the padding that it uses. Message authentication codes define their own padding, so this default does not apply to authentication transforms.

The OPTIONAL MKI and the RECOMMENDED authentication tag are the only fields defined by SRTP that are not in RTP. Only 8-bit alignment is assumed.

MKI (Master Key Identifier): configurable length, OPTIONAL
: The MKI is defined, signaled, and used by key management. The MKI identifies the master key from which the session key(s) were derived that authenticate and/or encrypt the particular packet. Note that the MKI SHALL NOT identify the SRTP cryptographic context, which is identified according to Section 3.2.3. The MKI MAY be used by key management for the purposes of re-keying, identifying a particular master key within the cryptographic context (Section 3.2.1).

Authentication tag: configurable length, RECOMMENDED
: The authentication tag is used to carry message authentication data. The Authenticated Portion of an SRTP packet consists of the RTP header followed by the Encrypted Portion of the SRTP packet. Thus, if both encryption and authentication are applied, encryption SHALL be applied before authentication on the sender side and conversely on the receiver side. The authentication tag provides authentication of the RTP header and payload, and it indirectly provides replay protection by authenticating the

sequence number. Note that the MKI is not integrity protected as this does not provide any extra protection.

3.2 SRTP Cryptographic Contexts

Each SRTP stream requires the sender and receiver to maintain cryptographic state information. This information is called the "cryptographic context".

SRTP uses two types of keys: session keys and master keys. By a "session key", we mean a key which is used directly in a cryptographic transform (e.g. encryption or message authentication), and by a "master key", we mean a random bit string (given by the key management protocol) from which session keys are derived in a cryptographically secure way. The master key(s) and other parameters in the cryptographic context are provided by key management mechanisms external to SRTP, see Section 8.

3.2.1 Transform-independent parameters

Transform-independent parameters are present in the cryptographic context independently of the particular encryption or authentication transforms that are used. The transform-independent parameters of the cryptographic context for SRTP consist of:

* a 32-bit unsigned rollover counter (ROC), which records how many times the 16-bit RTP sequence number has been reset to zero after passing through 65,535. Unlike the sequence number (SEQ), which SRTP extracts from the RTP packet header, the ROC is maintained by SRTP as described in Section 3.3.1.

  We define the index of the SRTP packet corresponding to a given ROC and RTP sequence number to be the 48-bit quantity

      i = 2^16 * ROC + SEQ.

* for the receiver only, a 16-bit sequence number s_l, which can be thought of as the highest received RTP sequence number (see Section 3.3.1 for its handling), which SHOULD be authenticated since message authentication is RECOMMENDED,

* an identifier for the encryption algorithm, i.e., the cipher and its mode of operation,

* an identifier for the message authentication algorithm,

* a replay list, maintained by the receiver only (when authentication and replay protection are provided), containing

indices of recently received and authenticated SRTP packets,

* an MKI indicator (0/1) as to whether an MKI is present in SRTP and SRTCP packets,

* if the MKI indicator is set to one, the length (in octets) of the MKI field, and (for the sender) the actual value of the currently active MKI (the value of the MKI indicator and length MUST be kept fixed for the lifetime of the context),

* the master key(s), which MUST be random and kept secret,

* for each master key, there is a counter of the number of SRTP packets that have been processed (sent) with that master key (essential for security, see Sections 3.3.1 and 9),

* non-negative integers n_e, and n_a, determining the length of the session keys for encryption, and message authentication.

In addition, for each master key, an SRTP stream MAY use the following associated values:

* a master salt, to be used in the key derivation of session keys. This value, when used, MUST be random, but MAY be public. Use of master salt is strongly RECOMMENDED, see Section 9.2. A "NULL" salt is treated as 00...0.

* an integer in the set {1,2,4,...,2^24}, the "key_derivation_rate", where an unspecified value is treated as zero. The constraint to be a power of 2 simplifies the session-key derivation implementation, see Section 4.3.

* an MKI value,

* <"From", "To"> values, specifying the lifetime for a master key, expressed in terms of the two 48-bit index values inside whose range (including the range end-points) the master key is valid. For the use of <From, To>, see Section 8.1.1. <"From", "To"> is an alternative to the MKI and assumes that a master key is in one-to-one correspondence with the SRTP session key on which the <"From", "To"> range is defined.

SRTCP SHALL by default share the crypto context with SRTP, except:

* no rollover counter and s_l-value need to be maintained as the RTCP index is explicitly carried in each SRTCP packet,

* a separate replay list is maintained (when replay protection is

provided),

* SRTCP maintains a separate counter for its master key (even if the master key is the same as that for SRTP, see below), as a means to maintain a count of the number of SRTCP packets that have been processed with that key.

Note in particular that the master key(s) MAY be shared between SRTP and the corresponding SRTCP, if the pre-defined transforms (including the key derivation) are used but the session key(s) MUST NOT be so shared.

In addition, there can be cases (see Sections 8 and 9.1) where several SRTP streams within a given RTP session, identified by their synchronization source (SSRCs, which is part of the RTP header), share most of the crypto context parameters (including possibly master and session keys). In such cases, just as in the normal SRTP/SRTCP parameter sharing above, separate replay lists and packet counters for each stream (SSRC) MUST still be maintained. Also, separate SRTP indices MUST then be maintained.

A summary of parameters, pre-defined transforms, and default values for the above parameters (and other SRTP parameters) can be found in Sections 5 and 8.2.

3.2.2 Transform-dependent parameters

All encryption, authentication/integrity, and key derivation parameters are defined in the transforms section (Section 4). Typical examples of such parameters are block size of ciphers, session keys, data for the Initialization Vector (IV) formation, etc. Future SRTP transform specifications MUST include a section to list the additional cryptographic context's parameters for that transform, if any.

3.2.3 Mapping SRTP Packets to Cryptographic Contexts

Recall that an RTP session for each participant is defined [RTPNEW] by a pair of destination transport addresses (one network address plus a port pair for RTP and RTCP), and that a multimedia session is defined as a collection of RTP sessions. For example, a particular multimedia session could include an audio RTP session, a video RTP session, and a text RTP session.

A cryptographic context SHALL be uniquely identified by the triplet context identifier:

context id = <SSRC, destination network address, destination transport port number>

where the destination network address and the destination transport port are the ones in the SRTP packet. It is assumed that, when presented with this information, the key management returns a context with the information as described in Section 3.2.

As noted above, SRTP and SRTCP by default share the bulk of the parameters in the cryptographic context. Thus, retrieving the crypto context parameters for an SRTCP stream in practice may imply a binding to the correspondent SRTP crypto context. It is up to the implementation to assure such binding, since the RTCP port may not be directly deducible from the RTP port only. Alternatively, the key management may choose to provide separate SRTP- and SRTCP-contexts, duplicating the common parameters (such as master key(s)). The latter approach then also enables SRTP and SRTCP to use, e.g., distinct transforms, if so desired. Similar considerations arise when multiple SRTP streams, forming part of one single RTP session, share keys and other parameters.

If no valid context can be found for a packet corresponding to a certain context identifier, that packet MUST be discarded from further processing.

3.3 SRTP Packet Processing

The following applies to SRTP. SRTCP is described in Section 3.4.

Assuming initialization of the cryptographic context(s) has taken place via key management, the sender SHALL do the following to construct an SRTP packet:

1. Determine which cryptographic context to use as described in Section 3.2.3.

2. Determine the index of the SRTP packet using the rollover counter, the highest sequence number in the cryptographic context, and the sequence number in the RTP packet, as described in Section 3.3.1.

3. Determine the master key and master salt. This is done using the index determined in the previous step or the current MKI in the cryptographic context, according to Section 8.1.

4. Determine the session keys and session salt (if they are used by the transform) as described in Section 4.3, using master key, master

salt, key_derivation_rate, and session key-lengths in the cryptographic context with the index, determined in Steps 2 and 3.

5. Encrypt the RTP payload to produce the Encrypted Portion of the packet (see Section 4.1, for the defined ciphers). This step uses the encryption algorithm indicated in the cryptographic context, the session encryption key and the session salt (if used) found in Step 4 together with the index found in Step 2.

6. If the MKI indicator is set to one, append the MKI to the packet.

7. For message authentication, compute the authentication tag for the Authenticated Portion of the packet, as described in Section 4.2. This step uses the current rollover counter, the authentication algorithm indicated in the cryptographic context, and the session authentication key found in Step 4. Append the authentication tag to the packet.

8. If necessary, update the ROC as in Section 3.3.1, using the packet index determined in Step 2.

To authenticate and decrypt an SRTP packet, the receiver SHALL do the following:

1. Determine which cryptographic context to use as described in Section 3.2.3.

2. Run the algorithm in Section 3.3.1 to get the index of the SRTP packet. The algorithm uses the rollover counter and highest sequence number in the cryptographic context with the sequence number in the SRTP packet, as described in Section 3.3.1.

3. Determine the master key and master salt. If the MKI indicator in the context is set to one, use the MKI in the SRTP packet, otherwise use the index from the previous step, according to Section 8.1.

4. Determine the session keys, and session salt (if used by the transform) as described in Section 4.3, using master key, master salt, key_derivation_rate and session key-lengths in the cryptographic context with the index, determined in Steps 2 and 3.

5. For message authentication and replay protection, first check if the packet has been replayed (Section 3.3.2), using the Replay List and the index as determined in Step 2. If the packet is judged to be replayed, then the packet MUST be discarded, and the event SHOULD be logged.

Next, perform verification of the authentication tag, using the rollover counter from Step 2, the authentication algorithm indicated in the cryptographic context, and the session authentication key from Step 4. If the result is "AUTHENTICATION FAILURE" (see Section 4.2), the packet MUST be discarded from further processing and the event SHOULD be logged.

6. Decrypt the Encrypted Portion of the packet (see Section 4.1, for the defined ciphers), using the decryption algorithm indicated in the cryptographic context, the session encryption key and salt (if used) found in Step 4 with the index from Step 2.

7. Update the rollover counter and highest sequence number, s_l, in the cryptographic context as in Section 3.3.1, using the packet index estimated in Step 2. If replay protection is provided, also update the Replay List as described in Section 3.3.2.

8. When present, remove the MKI and authentication tag fields from the packet.

3.3.1 Packet Index Determination, and ROC, s_l Update

SRTP implementations use an "implicit" packet index for sequencing, i.e., not all of the index is explicitly carried in the SRTP packet. For the pre-defined transforms, the index i is used in replay protection (Section 3.3.2), encryption (Section 4.1), message authentication (Section 4.2), and for the key derivation (Section 4.3).

When the session starts, the sender side MUST set the rollover counter, ROC, to zero. Each time the RTP sequence number, SEQ, wraps modulo 2^16, the sender side MUST increment ROC by one, modulo 2^32 (see security aspects below). The sender's packet index is then defined as

$$i = 2^{16} * ROC + SEQ.$$

Receiver-side implementations use the RTP sequence number to determine the correct index of a packet, which is the location of the packet in the sequence of all SRTP packets. A robust approach for the proper use of a rollover counter requires its handling and use to be well defined. In particular, out-of-order RTP packets with sequence numbers close to 2^16 or zero must be properly handled.

The index estimate is based on the receiver's locally maintained ROC and s_l values. At the setup of the session, the ROC MUST be set to zero. Receivers joining an on-going session MUST be given the

current ROC value using out-of-band signaling such as key-management signaling. Furthermore, the receiver SHALL initialize s_l to the RTP sequence number (SEQ) of the first observed SRTP packet (unless the initial value is provided by out of band signaling such as key management).

On consecutive SRTP packets, the receiver SHOULD estimate the index as

$$i = 2^{16} * v + SEQ,$$

where v is chosen from the set { ROC-1, ROC, ROC+1 } (modulo 2^32) such that i is closest (in modulo 2^48 sense) to the value 2^16 * ROC + s_l (see Appendix A for pseudocode).

After the packet has been processed and authenticated (when enabled for SRTP for SRTP packets for the session), the receiver MUST use v to conditionally update its s_l and ROC variables as follows. If v=(ROC-1) mod 2^32, then there is no update to s_l or ROC. If v=ROC, then s_l is set to SEQ if and only if SEQ is larger than the current s_l; there is no change to ROC. If v=(ROC+1) mod 2^32, then s_l is set to SEQ and ROC is set to v.

After a re-keying occurs (changing to a new master key), the rollover counter always maintains its sequence of values, i.e., it MUST NOT be reset to zero.

As the rollover counter is 32 bits long and the sequence number is 16 bits long, the maximum number of packets belonging to a given SRTP stream that can be secured with the same key is 2^48 using the pre-defined transforms. After that number of SRTP packets have been sent with a given (master or session) key, the sender MUST NOT send any more packets with that key. (There exists a similar limit for SRTCP, which in practice may be more restrictive, see Section 9.2.) This limitation enforces a security benefit by providing an upper bound on the amount of traffic that can pass before cryptographic keys are changed. Re-keying (see Section 8.1) MUST be triggered, before this amount of traffic, and MAY be triggered earlier, e.g., for increased security and access control to media. Recurring key derivation by means of a non-zero key_derivation_rate (see Section 4.3), also gives stronger security but does not change the above absolute maximum value.

On the receiver side, there is a caveat to updating s_l and ROC: if message authentication is not present, neither the initialization of s_l, nor the ROC update can be made completely robust. The receiver's "implicit index" approach works for the pre-defined transforms as long as the reorder and loss of the packets are not too great and bit-errors do not occur in unfortunate ways. In

particular, 2^15 packets would need to be lost, or a packet would need to be 2^15 packets out of sequence before synchronization is lost. Such drastic loss or reorder is likely to disrupt the RTP application itself.

The algorithm for the index estimate and ROC update is a matter of implementation, and should take into consideration the environment (e.g., packet loss rate) and the cases when synchronization is likely to be lost, e.g. when the initial sequence number (randomly chosen by RTP) is not known in advance (not sent in the key management protocol) but may be near to wrap modulo 2^16.

A more elaborate and more robust scheme than the one given above is the handling of RTP's own "rollover counter", see Appendix A.1 of [RTPNEW].

3.3.2 Replay Protection

Secure replay protection is only possible when integrity protection is present. It is RECOMMENDED to use replay protection, both for RTP and RTCP, as integrity protection alone cannot assure security against replay attacks.

A packet is "replayed" when it is stored by an adversary, and then re-injected into the network. When message authentication is provided, SRTP protects against such attacks through a Replay List. Each SRTP receiver maintains a Replay List, which conceptually contains the indices of all of the packets which have been received and authenticated. In practice, the list can use a "sliding window" approach, so that a fixed amount of storage suffices for replay protection. Packet indices which lag behind the packet index in the context by more than SRTP-WINDOW-SIZE can be assumed to have been received, where SRTP-WINDOW-SIZE is a receiver-side, implementation-dependent parameter and MUST be at least 64, but which MAY be set to a higher value.

The receiver checks the index of an incoming packet against the replay list and the window. Only packets with index ahead of the window, or, inside the window but not already received, SHALL be accepted.

After the packet has been authenticated (if necessary the window is first moved ahead), the replay list SHALL be updated with the new index.

The Replay List can be efficiently implemented by using a bitmap to represent which packets have been received, as described in the Security Architecture for IP [RFC2401].

3.4 Secure RTCP

Secure RTCP follows the definition of Secure RTP. SRTCP adds three mandatory new fields (the SRTCP index, an "encrypt-flag", and the authentication tag) and one optional field (the MKI) to the RTCP packet definition. The three mandatory fields MUST be appended to an RTCP packet in order to form an equivalent SRTCP packet. The added fields follow any other profile-specific extensions.

According to Section 6.1 of [RTPNEW], there is a REQUIRED packet format for compound packets. SRTCP MUST be given packets according to that requirement in the sense that the first part MUST be a sender report or a receiver report. However, the RTCP encryption prefix (a random 32-bit quantity) specified in that Section MUST NOT be used since, as is stated there, it is only applicable to the encryption method specified in [RTPNEW] and is not needed by the cryptographic mechanisms used in SRTP.

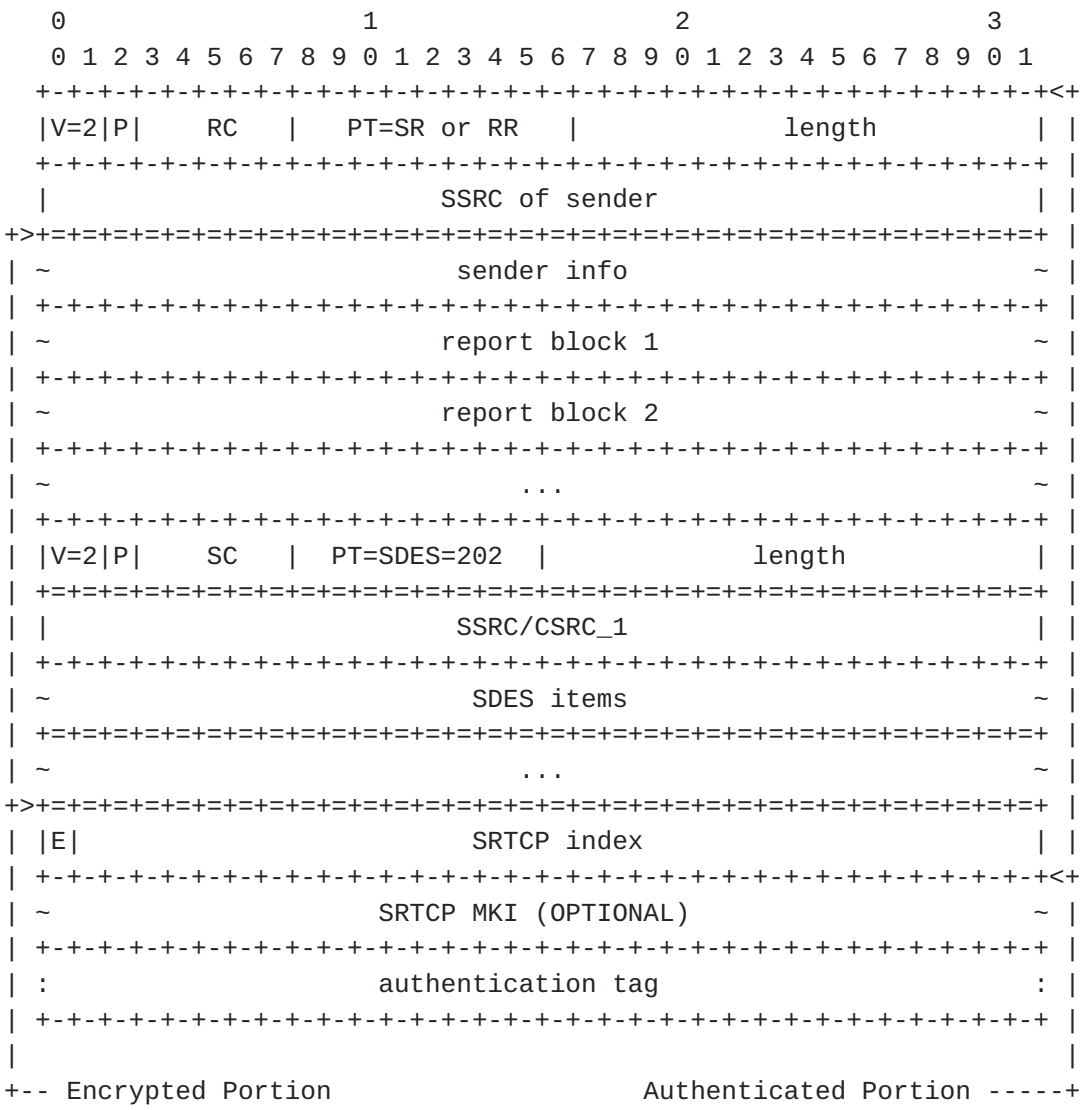

Figure 2. An example of the format of a Secure RTCP packet, consisting of an underlying RTCP compound packet with a Sender Report and SDES packet.

The Encrypted Portion of an SRTCP packet consists of the encryption (Section 4.1) of the RTCP payload of the equivalent compound RTCP packet, from the first RTCP packet, i.e., from the ninth (9) octet to the end of the compound packet. The Authenticated Portion of an SRTCP packet consists of the entire equivalent (eventually compound) RTCP packet, the E flag, and the SRTCP index (after any encryption has been applied to the payload).

The added fields are:

E-flag: 1 bit, REQUIRED
The E-flag indicates if the current SRTCP packet is encrypted or unencrypted. Section 9.1 of [RTPNEW] allows the split of a compound RTCP packet into two lower-layer packets, one to be encrypted and one to be sent in the clear. The E bit set to "1" indicates encrypted packet, and "0" indicates non-encrypted packet.

SRTCP index: 31 bits, REQUIRED
The SRTCP index is a 31-bit counter for the SRTCP packet. The index is explicitly included in each packet, in contrast to the "implicit" index approach used for SRTP. The SRTCP index MUST be set to zero before the first SRTCP packet is sent, and MUST be incremented by one, modulo 2^31, after each SRTCP packet is sent. In particular, after a re-key, the SRTCP index MUST NOT be reset to zero again (Section 3.3.1).

Authentication Tag: configurable length, REQUIRED
The authentication tag is used to carry message authentication data.

MKI: configurable length, OPTIONAL
The MKI is the Master Key Indicator, and functions according to the MKI definition in Section 3.

SRTCP uses the cryptographic context parameters and packet processing of SRTP by default, with the following changes:

* The receiver does not need to "estimate" the index, as it is explicitly signaled in the packet.

* Pre-defined SRTCP encryption is as specified in Section 4.1, but using the definition of the SRTCP Encrypted Portion given in this section, and using the SRTCP index as the index i. The encryption transform and related parameters SHALL by default be the same selected for the protection of the associated SRTP stream(s), while the NULL algorithm SHALL be applied to the RTCP packets not to be encrypted. SRTCP may have a different encryption transform than the one used by the corresponding SRTP. The expected use for this feature is when the former has NULL-encryption and the latter has a non NULL-encryption.

The E-flag is assigned a value by the sender depending on whether the packet was encrypted or not.

* SRTCP decryption is performed as in Section 4, but only if the E flag is equal to 1. If so, the Encrypted Portion is decrypted,

using the SRTCP index as the index i. In case the E-flag is 0, the payload is simply left unmodified.

* SRTCP replay protection is as defined in Section 3.3.2, but using the SRTCP index as the index i and a separate Replay List that is specific to SRTCP.

* The pre-defined SRTCP authentication tag is specified as in Section 4.2, but with the Authenticated Portion of the SRTCP packet given in this section (which includes the index). The authentication transform and related parameters (e.g., key size) SHALL by default be the same as selected for the protection of the associated SRTP stream(s)).

* In the last step of the processing, only the sender needs to update the value of the SRTCP index by incrementing it modulo 2^31 and for security reasons the sender MUST also check the number of SRTCP packets processed, see Section 9.2.

Message authentication for RTCP is REQUIRED, as it is the control protocol (e.g., it has a BYE packet) for RTP.

Precautions must be taken so that the packet expansion in SRTCP (due to the added fields) does not cause SRTCP messages to use more than their share of RTCP bandwidth. To avoid this, the following two measures MUST be taken:

1. When initializing the RTCP variable "avg_rtcp_size" defined in chapter 6.3 of [RTPNEW], it MUST include the size of the fields that will be added by SRTCP (index, E-bit, authentication tag, and when present, the MKI).

2. When updating the "avg_rtcp_size" using the variable packet_size" (section 6.3.3 of [RTPNEW]), the value of "packet_size" MUST include the size of the additional fields added by SRTCP.

With these measures in place the SRTCP messages will not use more than the allotted bandwidth. The effect of the size of the added fields on the SRTCP traffic will be that messages will be sent with longer packet intervals. The increase in the intervals will be directly proportional to size of the added fields. For the pre-defined transforms, the size of the added fields will be at least 14 octets, and upper bounded depending on MKI and the authentication tag sizes.

4. Pre-Defined Cryptographic Transforms

While there are numerous encryption and message authentication

algorithms that can be used in SRTP, we define below default algorithms in order to avoid the complexity of specifying the encodings for the signaling of algorithm and parameter identifiers. The defined algorithms have been chosen as they fulfill the goals listed in Section 2. Recommendations on how to extend SRTP with new transforms are given in Section 6.

4.1 Encryption

The following parameters are common to both pre-defined, non-NULL, encryption transforms specified in this section.

* BLOCK_CIPHER-MODE indicates the block cipher used and its mode of operation
* n_b is the bit-size of the block for the block cipher
* k_e is the session encryption key
* n_e is the bit-length of k_e
* k_s is the session salting key
* n_s is the bit-length of k_s
* SRTP_PREFIX_LENGTH is the octet length of the keystream prefix, an non-negative integer, specified by the message authentication code in use.

The distinct session keys and salts for SRTP/SRTCP are by default derived as specified in Section 4.3.

The encryption transforms defined in SRTP map the SRTP packet index and secret key into a pseudo-random keystream segment. Each keystream segment encrypts a single RTP packet. The process of encrypting a packet consists of generating the keystream segment corresponding to the packet, and then bitwise exclusive-oring that keystream segment onto the payload of the RTP packet to produce the Encrypted Portion of the SRTP packet. In case the payload size is not an integer multiple of n_b bits, the excess (least significant) bits of the keystream are simply discarded. Decryption is done the same way, but swapping the roles of the plaintext and ciphertext.

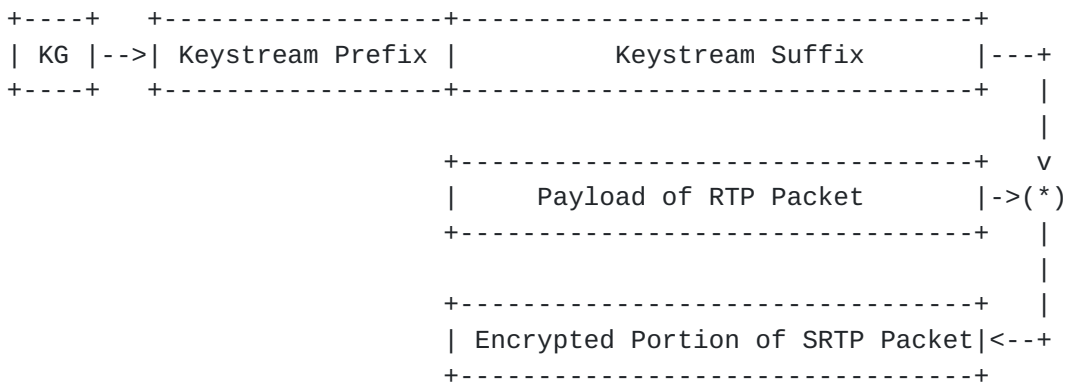

Figure 3: Default SRTP Encryption Processing. Here KG denotes the keystream generator, and (*) denotes bitwise exclusive-or.

The definition of how the keystream is generated, given the index, depends on the cipher and its mode of operation. Below, two such keystream generators are defined. The NULL cipher is also defined, to be used when encryption of RTP is not required.

The SRTP definition of the keystream is illustrated in Figure 3. The initial octets of each keystream segment MAY be reserved for use in a message authentication code, in which case the keystream used for encryption starts immediately after the last reserved octet. The initial reserved octets are called the "keystream prefix" (not to be confused with the "encryption prefix" of [RTPNEW, Section 6.1]), and the remaining octets are called the "keystream suffix". The keystream prefix MUST NOT be used for encryption. The process is illustrated in Figure 3.

The number of octets in the keystream prefix is denoted as SRTP_PREFIX_LENGTH. The keystream prefix is indicated by a positive, non-zero value of SRTP_PREFIX_LENGTH. This means that, even if confidentiality is not to be provided, the keystream generator output may still need to be computed for packet authentication, in which case the default keystream generator (mode) SHALL be used.

The default cipher is the Advanced Encryption Standard (AES), and we define two modes of running AES, (1) Segmented Integer Counter Mode AES and (2) AES in f8-mode. In the remainder of this section, let E(k,x) be AES applied to key k and input block x.

4.1.1 AES in Counter Mode

Conceptually, counter mode [AES-CTR] consists of encrypting successive integers. The actual definition is somewhat more

complicated, in order to randomize the starting point of the integer sequence. Each packet is encrypted with a distinct keystream segment, which SHALL be computed as follows.

A keystream segment SHALL be the concatenation of the 128-bit output blocks of the AES cipher in the encrypt direction, using key k = k_e, in which the block indices are in increasing order. Symbolically, each keystream segment looks like

E(k, IV) || E(k, IV + 1 mod 2^128) || E(k, IV + 2 mod 2^128) ...

where the 128-bit integer value IV SHALL be defined by the SSRC, the SRTP packet index i, and the SRTP session salting key k_s, as below.

IV = (k_s * 2^16) XOR (SSRC * 2^64) XOR (i * 2^16)

Each of the three terms in the XOR-sum above is padded with as many leading zeros as needed to make the operation well-defined, considered as a 128-bit value.

The inclusion of the SSRC allows the use of the same key to protect distinct SRTP streams within the same RTP session, see the security caveats in Section 9.1.

In the case of SRTCP, the SSRC of the first header of the compound packet MUST be used, i SHALL be the 31-bit SRTCP index and k_e, k_s SHALL be replaced by the SRTCP session key and salt.

Note that the initial value, IV, is fixed for each packet and is formed by "reserving" 16 zeros in the least significant bits for the purpose of the counter. The number of blocks of keystream generated for any fixed value of IV MUST NOT exceed 2^16 to avoid key stream re-use, see below. The AES has a block size of 128 bits, so 2^16 output blocks are sufficient to generate the 2^23 bits of keystream needed to encrypt the largest possible RTP packet (except for IPv6 "jumbograms" [RFC2675], which are not likely to be used for RTP-based multimedia traffic). This restriction on the maximum bit-size of the packet that can be encrypted ensures the security of the encryption method by limiting the effectiveness of probabilistic attacks [BDJR].

For a particular Counter Mode key, each IV value used as an input MUST be distinct, in order to avoid the security exposure of a two-time pad situation (Section 9.1). To satisfy this constraint, an implementation MUST ensure that the values of the SRTP packet index of ROC || SEQ, and the SSRC used in the construction of the IV are distinct for any particular key. The failure to ensure this uniqueness could be catastrophic for Secure RTP. This is in

contrast to the situation for RTP itself, which may be able to tolerate such failures. It is RECOMMENDED that, if a dedicated security module is present, the RTP sequence numbers and SSRC either be generated or checked by that module (i.e., sequence-number and SSRC processing in an SRTP system needs to be protected as well as the key).

4.1.2 AES in f8-mode

To encrypt UMTS (Universal Mobile Telecommunications System, as 3G networks) data, a solution (see [f8-a], [f8-b]) known as the f8-algorithm has been developed. On a high level, the proposed scheme is a variant of Output Feedback Mode (OFB) [HAC], with a more elaborate initialization and feedback function. As in normal OFB, the core consists of a block cipher. We also define here the use of AES as a block cipher to be used in what we shall call "f8-mode of operation" RTP encryption. The AES f8-mode SHALL use the same default sizes for session key and salt as AES counter mode.

Figure 4 shows the structure of block cipher, E, running in f8-mode.

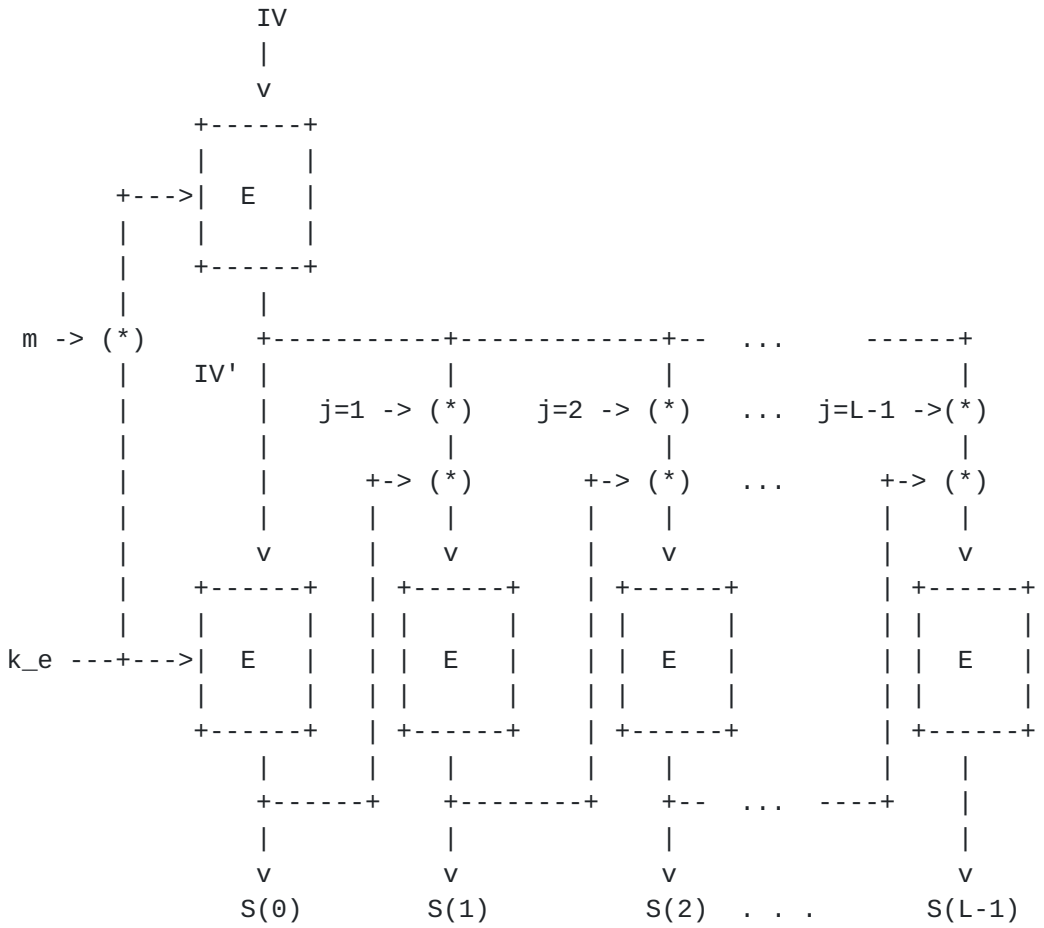

Figure 4. f8-mode of operation (asterisk, (*), denotes bitwise XOR). The figure represents the KG in Figure 3, when AES-f8 is used.

4.1.2.1 f8 Keystream Generation

The Initialization Vector (IV) SHALL be determined as described in Section 4.1.2.2 (and in Section 4.1.2.3 for SRTCP).

Let IV', S(j), and m denote n_b-bit blocks. The keystream, S(0) || ... || S(L-1), for an N-bit message SHALL be defined by setting IV' = E(k_e XOR m, IV), and S(-1) = 00..0. For j = 0,1,..,L-1 where L = N/n_b (rounded up to nearest integer) compute

$$S(j) = E(k_e, IV' \text{ XOR } j \text{ XOR } S(j-1))$$

Notice that the IV is not used directly. Instead it is fed through E under another key to produce an internal, "masked" value (denoted IV') to prevent an attacker from gaining known input/output pairs.

The role of the internal counter, j, is to prevent short keystream cycles. The value of the key mask m SHALL be

m = k_s || 0x555..5,

i.e. the session salting key, appended by the binary pattern 0101.. to fill out the entire desired key size, n_e.

The sender SHOULD NOT generate more than 2^32 blocks, which is sufficient to generate 2^39 bits of keystream. Unlike counter mode, there is no absolute threshold above (below) which f8 is guaranteed to be insecure (secure). The above bound has been chosen to limit, with sufficient security margin, the probability of degenerative behavior in the f8 keystream generation.

4.1.2.2 f8 SRTP IV Formation

The purpose of the following IV formation is to provide a feature which we call implicit header authentication (IHA), see Section 9.5.

The SRTP IV for 128-bit block AES-f8 SHALL be formed in the following way:

IV = 0x00 || M || PT || SEQ || TS || SSRC || ROC

M, PT, SEQ, TS, SSRC SHALL be taken from the RTP header; ROC is from the cryptographic context.

The presence of the SSRC as part of the IV allows AES-f8 to be used when a master key is shared between multiple streams within the same RTP session, see Section 9.1.

4.1.2.3 f8 SRTCP IV Formation

The SRTCP IV for 128-bit block AES-f8 SHALL be formed in the following way:

IV= 0..0 || E || SRTCP index || V || P || RC || PT || length || SSRC

where V, P, RC, PT, length, SSRC SHALL be taken from the first header in the RTCP compound packet. E and SRTCP index are the 1-bit and 31-bit fields added to the packet.

4.1.3 NULL Cipher

The NULL cipher is used when no confidentiality for RTP/RTCP is requested. The keystream can be thought of as "000..0", i.e. the

encryption SHALL simply copy the plaintext input into the ciphertext output.

4.2 Message Authentication and Integrity

Throughout this section, M will denote data to be integrity protected. In the case of SRTP, M SHALL consist of the Authenticated Portion of the packet (as specified in Figure 1) concatenated with the ROC, M = Authenticated Portion || ROC; in the case of SRTCP, M SHALL consist of the Authenticated Portion (as specified in Figure 2) only.

Common parameters:

* AUTH_ALG is the authentication algorithm
* k_a is the session message authentication key
* n_a is the bit-length of the authentication key
* n_tag is the bit-length of the output authentication tag
* SRTP_PREFIX_LENGTH is the octet length of the keystream prefix as defined above, a parameter of AUTH_ALG

The distinct session authentication keys for SRTP/SRTCP are by default derived as specified in Section 4.3.

The values of n_a, n_tag, and SRTP_PREFIX_LENGTH MUST be fixed for any particular fixed value of the key.

We describe the process of computing authentication tags as follows. The sender computes the tag of M and appends it to the packet. The SRTP receiver verifies a message/authentication tag pair by computing a new authentication tag over M using the selected algorithm and key, and then compares it to the tag associated with the received message. If the two tags are equal, then the message/tag pair is valid; otherwise, it is invalid and the error audit message "AUTHENTICATION FAILURE" MUST be returned.

4.2.1 HMAC-SHA1

The pre-defined authentication transform for SRTP is HMAC-SHA1 [RFC2104]. With HMAC-SHA1, the SRTP_PREFIX_LENGTH (Figure 3) SHALL be 0. For SRTP (respectively SRTCP), the HMAC SHALL be applied to the session authentication key and M as specified above, i.e. HMAC(k_a, M). The HMAC output SHALL then be truncated to the n_tag left-most bits.

4.3 Key Derivation

4.3.1 Key Derivation Algorithm

Regardless of the encryption or message authentication transform that is employed (it may be an SRTP pre-defined transform or newly introduced according to Section 6), interoperable SRTP implementations MUST use the SRTP key derivation to generate session keys. Once the key derivation rate is properly signaled at the start of the session, there is no need for extra communication between the parties that use SRTP key derivation.

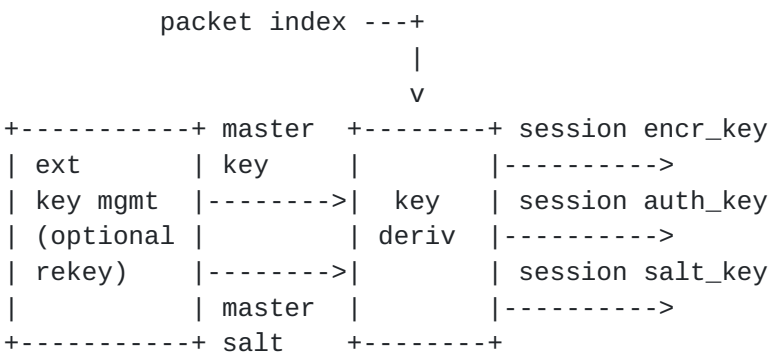

Figure 5: SRTP key derivation.

At least one initial key derivation SHALL be performed by SRTP, i.e., the first key derivation is REQUIRED. Further applications of the key derivation MAY be performed, according to the "key_derivation_rate" value in the cryptographic context. The key derivation function SHALL be initially invoked before the first packet and then, if derivation rate is r > 0, further invoked on every r-th packet, and produce session keys according to the non-zero key derivation rate. This can be thought of as "refreshing" the session keys. The value of "key_derivation_rate" MUST be kept fixed for the lifetime of the associated master key.

Interoperable SRTP implementations MAY also derive session salting keys for encryption transforms, as is done in both of the pre-defined transforms.

Let m and n be positive integers. A pseudo-random function family is a set of keyed functions {PRF_n(k,x)} such that for the (secret) random key k, given m-bit x, PRF_n(k,x) is an n-bit string, computationally indistinguishable from random n-bit strings, see [HAC]. For the purpose of key derivation in SRTP, a secure PRF with m = 128 (or more) MUST be used, and a default PRF transform is defined in Section 4.3.3.

Let "a DIV t" denote integer division of a by t, rounded down, and with the convention that "a DIV 0 = 0" for all a. We also make the convention of treating "a DIV t" as a bit string of the same length as a, and thus "a DIV t" will in general have leading zeros.

Key derivation SHALL be defined as follows in terms of <label>, an 8-bit constant (see below), master_salt and key_derivation_rate, as determined in the cryptographic context, and index, the packet index (i.e., the 48-bit ROC || SEQ for SRTP):

* Let r = index DIV key_derivation_rate (with DIV as defined above).

* Let key_id = <label> || r.

* Let x = key_id XOR master_salt, where key_id and master_salt are aligned so that their least significant bits agree (right-alignment).

<label> MUST be unique for each type of key to be derived. We currently define <label> 0x00 to 0x05 (see below), and future extensions MAY specify new values in the range 0x06 to 0xff for other purposes. The n-bit SRTP key (or salt) for this packet SHALL then be derived from the master key, k_master as follows:

PRF_n(k_master, x).

(The PRF may internally specify additional formatting and padding of x, see e.g. Section 4.3.3 for the default PRF.)

The session keys and salt SHALL now be derived using:

- k_e (SRTP encryption): <label> = 0x00, n = n_e.

- k_a (SRTP message authentication): <label> = 0x01, n = n_a.

- k_s (SRTP salting key) <label> = 0x02, n = n_s.

where n_e, n_s, and n_a are from the cryptographic context.

The master key and master salting key MUST be random, but the master salt MAY be public.

Note that for a key_derivation_rate of 0, the application of the key derivation SHALL take place exactly once.

The definition of DIV above is purely for notational convenience. For a non-zero t among the set of allowed key derivation rates, "a

DIV t" can be implemented as a right-shift by the base-2 logarithm of t. The derivation operation is further facilitated if the rates are chosen to be powers of 256, but that granularity was considered too coarse to be a requirement of this specification.

The upper limit on the number of packets that can be secured using the same master key (see Section 9.2) is independent of the key derivation.

4.3.2 SRTCP Key Derivation

SRTCP SHALL by default use the same master key (and master salt) as SRTP. To do this securely, the following changes SHALL be done to the definitions in Section 4.3.1 when applying session key derivation for SRTCP.

Replace the SRTP index by the 32-bit quantity: 0 || SRTCP index (i.e. excluding the E-bit, replacing it with a fixed 0-bit), and use <label> = 0x03 for the SRTCP encryption key, <label> = 0x04 for the SRTCP authentication key, and, <label> = 0x05 for the SRTCP salting key.

4.3.3 AES-CM PRF

The currently defined PRF, keyed by 128, 192, or 256 bit master key, has input block size m = 128 and can produce n-bit outputs for n up to 2^23. PRF_n(k_master,x) SHALL be AES in Counter Mode as described in Section 4.1.1, applied to key k_master, and IV equal to (x*2^16), and with the output keystream truncated to the n first (left-most) bits. (Requiring n/128, rounded up, applications of AES.)

5. Default and mandatory-to-implement Transforms

The default transforms also are mandatory-to-implement transforms in SRTP. Of course, "mandatory-to-implement" does not imply "mandatory-to-use". Table 1 summarizes the pre-defined transforms. The default values below are valid for the pre-defined transforms.

	mandatory-to-impl.	optional	default
encryption	AES-CM, NULL	AES-f8	AES-CM
message integrity	HMAC-SHA1	-	HMAC-SHA1
key derivation (PRF)	AES-CM	-	AES-CM

Table 1: Mandatory-to-implement, optional and default transforms in SRTP and SRTCP.

5.1 Encryption: AES-CM and NULL

AES running in Segmented Integer Counter Mode, as defined in Section 4.1.1, SHALL be the default encryption algorithm. The default key lengths SHALL be 128-bit for the session encryption key (n_e). The default session salt key-length (n_s) SHALL be 112 bits.

The NULL cipher SHALL also be mandatory-to-implement.

5.2 Message Authentication/Integrity: HMAC-SHA1

HMAC-SHA1, as defined in Section 4.2.1, SHALL be the default message authentication code. The default session authentication key-length (n_a) SHALL be 160 bits, the default authentication tag length (n_tag) SHALL be 80 bits, and the SRTP_PREFIX_LENGTH SHALL be zero for HMAC-SHA1. In addition, for SRTCP, the pre-defined HMAC-SHA1 MUST NOT be applied with a value of n_tag, nor n_a, that are smaller than these defaults. For SRTP, smaller values are NOT RECOMMENDED, but MAY be used after careful consideration of the issues in Section 7.5 and 9.5.

5.3 Key Derivation: AES-CM PRF

The AES Counter Mode based key derivation and PRF defined in Sections 4.3.1 to 4.3.3, using a 128-bit master key, SHALL be the default method for generating session keys. The default master salt length SHALL be 112 bits and the default key-derivation rate SHALL be zero.

6. Adding SRTP Transforms

Section 4 provides examples of the level of detail needed for defining transforms. Whenever a new transform is to be added to SRTP, a companion standard track RFC MUST be written to exactly define how the new transform can be used with SRTP (and SRTCP). Such a companion RFC SHOULD avoid overlap with the SRTP protocol document. Note however, that it MAY be necessary to extend the SRTP or SRTCP cryptographic context definition with new parameters

(including fixed or default values), add steps to the packet processing, or even add fields to the SRTP packets. The companion RFC SHALL explain any known issues regarding interactions between the transform and other aspects of SRTP.

Each new transform document SHOULD specify its key attributes, e.g., size of keys (minimum, maximum, recommended), format of keys, recommended/required processing of input keying material, requirements/recommendations on key lifetime, re-keying and key derivation, whether sharing of keys between SRTP and SRTCP is allowed or not, etc.

An added message integrity transform SHOULD define a minimum acceptable key/tag size for SRTCP, equivalent in strength to the minimum values as defined in Section 5.2.

7. Rationale

This section explains the rationale behind several important features of SRTP.

7.1 Key derivation

Key derivation reduces the burden on the key establishment. As many as six different keys are needed per crypto context (SRTP and SRTCP encryption keys and salts, SRTP and SRTCP authentication keys), but these are derived from a single master key in a cryptographically secure way. Thus, the key management protocol needs to exchange only one master key (plus master salt when required), and then SRTP itself derives all the necessary session keys (via the first, mandatory application of the key derivation function).

Multiple applications of the key derivation function are optional, but will give security benefits when enabled. They prevent an attacker from obtaining large amounts of ciphertext produced by a single fixed session key. If the attacker was able to collect a large amount of ciphertext for a certain session key, he might be helped in mounting certain attacks.

Multiple applications of the key derivation function provide backwards and forward security in the sense that a compromised session key does not compromise other session keys derived from the same master key. This means that the attacker who is able to recover a certain session key, is anyway not able to have access to messages secured under previous and later session keys (derived from the same master key). (Note that, of course, a leaked master key reveals all the session keys derived from it.)

Considerations arise with high-rate key refresh, especially in large multicast settings, see Section 11.

7.2 Salting key

The master salt guarantees security against off-line key-collision attacks on the key derivation that might otherwise reduce the effective key size [MF00].

The derived session salting key used in the encryption, has been introduced to protect against some attacks on additive stream ciphers, see Section 9.2. The explicit inclusion method of the salt in the IV has been selected for ease of hardware implementation.

7.3 Message Integrity from Universal Hashing

The particular definition of the keystream given in Section 4.1 (the keystream prefix) is to give provision for particular universal hash functions, suitable for message authentication in the Wegman-Carter paradigm [WC81]. Such functions are provably secure, simple, quick, and especially appropriate for Digital Signal Processors and other processors with a fast multiply operation.

No authentication transforms are currently provided in SRTP other than HMAC-SHA1. Future transforms, like the above mentioned universal hash functions, MAY be added following the guidelines in Section 6.

7.4 Data Origin Authentication Considerations

Note that in pair-wise communications, integrity and data origin authentication are provided together. However, in group scenarios where the keys are shared between members, the MAC tag only proves that a member of the group sent the packet, but does not prevent against a member impersonating another. Data origin authentication (DOA) for multicast and group RTP sessions is a hard problem that needs a solution; while some promising proposals are being investigated [PCST1, PCST2], more work is needed to rigorously specify these technologies. Thus SRTP data origin authentication in groups is for further study.

DOA can be done otherwise using signatures. However, this has high impact in terms of bandwidth and processing time, therefore we do not offer this form of authentication in the pre-defined packet-integrity transform.

The presence of mixers and translators does not allow data origin authentication in case the RTP payload and/or the RTP header are

manipulated. Note that these types of middle entities also disrupt end-to-end confidentiality (as the IV formation depends e.g. on the RTP header preservation). A certain trust model may choose to trust the mixers/translators to decrypt/re-encrypt the media (this would imply breaking the end-to-end security, with related security implications).

7.5 Short and Zero-length Message Authentication

As shown in Figure 1, the authentication tag is RECOMMENDED in SRTP. A full 80-bit authentication-tag SHOULD be used, but a shorter tag or even a zero-length tag (i.e. no message authentication) MAY be used under certain conditions to support either of the following two application environments.

1. Strong authentication can be impractical in environments where bandwidth preservation is imperative. An important special case is wireless communication systems, in which bandwidth is a scarce and expensive resource. Studies have shown that for certain applications and link technologies, additional bytes may result in a significant decrease in spectrum efficiency [SWO]. Considerable effort has been made to design IP header compression techniques to improve spectrum efficiency [ROHC]. A typical voice application produces 20 byte samples, and the RTP, UDP and IP headers need to be jointly compressed to one or two bytes on average in order to obtain acceptable wireless bandwidth economy [RFC3095]. In this case, strong authentication would impose nearly fifty percent overhead.

2. Authentication is impractical for applications that use data links with fixed-width fields that cannot accommodate the expansion due to the authentication tag. This is the case for some important existing wireless channels. For example, zero-byte header compression is used to adapt EVRC/SMV voice with the legacy IS-95 bearer channel in CDMA2000 VoIP services. It was found that a not a single additional octet could be added to the data, which motivated the creation of a zero-byte profile for ROHC [RFC3242].

A short tag is secure for a restricted set of applications. Consider a voice telephony application, for example, such as a G.729 audio codec with a 20-millisecond packetization interval, protected by a 32-bit message authentication tag. The likelihood of any given packet being successfully forged is only one in 2^32. Thus an adversary can control no more than 20 milliseconds of audio output during a 994-day period, on average. In contrast, the effect of a single forged packet can be much larger if the application is stateful. A codec that uses relative or predictive compression

across packets will propagate the maliciously generated state, affecting a longer duration of output.

Certainly not all SRTP or telephony applications meet the criteria for short or zero-length authentication tags. Section 9.5.1 discusses the risks of weak or no message authentication, and section 9.5 describes the circumstances when it is acceptable and when it is unacceptable.

8. Key Management Considerations

There are emerging key management standards [MIKEY, KEYMGT, SDMS] for establishing an SRTP cryptographic context (e.g. an SRTP master key). Both proprietary and open-standard key management methods are likely to be used for telephony applications [MIKEY, KINK] and multicast applications [GDOI]. This section provides guidance for key management systems that service SRTP session.

For initialization, an interoperable SRTP implementation SHOULD be given the SSRC and MAY be given the initial RTP sequence number for the RTP stream by key management (thus, key management has a dependency on RTP operational parameters). Sending the RTP sequence number in the key management may be useful e.g. when the initial sequence number is close to wrapping (to avoid synchronization problems), and to communicate the current sequence number to a joining endpoint (to properly initialize its replay list).

If the pre-defined transforms are used, SRTP allows sharing of the same master key between SRTP/SRTCP streams belonging to the same RTP session.

First, sharing between SRTP streams belonging to the same RTP session is secure if the design of the synchronization mechanism, i.e., the IV, avoids keystream re-use (the two-time pad, Section 9.1). This is taken care of by the fact that RTP provides for unique SSRCs for streams belonging to the same RTP session. See Section 9.1 for further discussion.

Second, sharing between SRTP and the corresponding SRTCP is secure. The fact that an SRTP stream and its associated SRTCP stream both carry the same SSRC does not constitute a problem for the two time pad due to the key derivation. Thus, SRTP and SRTCP corresponding to one RTP session MAY share master keys (as they do by default).

Note that also message authentication has a dependency on SSRC uniqueness that is unrelated to the problem of keystream reuse: SRTP streams authenticated under the same key MUST have a distinct SSRC in order to identify the sender of the message. This requirement is

needed because the SSRC is the cryptographically authenticated field used to distinguish between different SRTP streams. Were two streams to use identical SSRC values, then an adversary could substitute messages from one stream into the other without detection.

SRTP/SRTCP MUST NOT share master keys under any other circumstances than the ones given above, i.e. between SRTP and its corresponding SRTCP, and, between streams belonging to the same RTP session.

8.1. Re-keying

The recommended way for a particular key management system to provide re-key within SRTP is by associating a master key in a crypto context with an MKI.

This provides for easy master key retrieval (see Scenarios in Section 11), but has the disadvantage of adding extra bits to each packet. As noted in Section 7.5, some wireless links do not cater for added bits, therefore SRTP also defines a more economic way of triggering re-keying, via use of <From, To>, which works in some specific, simple scenarios (see Section 8.1.1).

SRTP senders SHALL count the amount of SRTP and SRTCP traffic being used for a master key and invoke key management to re-key if needed (Section 9.2). These interactions are defined by the key management interface to SRTP and are not defined by this protocol specification.

8.1.1 Use of the <From, To> for re-keying

In addition to the use of the MKI, SRTP defines another optional mechanism for master key retrieval, the <From, To>. The <From, To> specifies the range of SRTP indices (a pair of sequence number and ROC) within which a certain master key is valid, and is (when used) part of the crypto context. By looking at the 48-bit SRTP index of the current SRTP packet, the corresponding master key can be found by determining which From-To interval it belongs to. For SRTCP, the most recently observed/used SRTP index (which can be obtained from the cryptographic context) is used for this purpose, even though SRTCP has its own (31-bit) index (see caveat below).

This method, compared to the MKI, has the advantage of identifying the master key and defining its lifetime without adding extra bits to each packet. This could be useful, as already noted, for some wireless links that do not cater for added bits. However, its use SHOULD be limited to specific, very simple scenarios. We recommend to limit its use when the RTP session is a simple unidirectional or

bi-directional stream. This is because in case of multiple streams, it is difficult to trigger the re-key based on the <From, To> of a single RTP stream. E.g., if several streams share a master key, there is no simple one-to-one correspondence between the index sequence space of a certain stream, and the index sequence space on which the <From, To> values are based. Consequently, when a master key is shared between streams, one of these streams MUST be designated by key management as the one whose index space defines the re-keying points. Also, the re-key triggering on SRTCP is based on the correspondent SRTP stream, i.e. when the SRTP stream changes the master key, so does the correspondent SRTCP. This becomes obviously more and more complex with multiple streams.

The default values for the <From, To> are "from the first observed packet" and "until further notice". However, the maximum limit of SRTP/SRTCP packets that are sent under each given master/session key (Section 9.2) MUST NOT be exceeded.

In case the <From, To> is used as key retrieval, then the MKI is not inserted in the packet (and its indicator in the crypto context is zero). However, using the MKI does not exclude using <"From", "To"> key lifetime simultaneously. This can for instance be useful to signal at the sender side at which point in time an MKI is to be made active.

8.2. Key Management parameters

The table below lists all SRTP parameters that key management can supply. For reference, it also provides a summary of the default and mandatory-to-support values for an SRTP implementation as described in Section 5.

Parameter	Mandatory-to-support	Default
SRTP and SRTCP encr transf. (Other possible values: AES_f8)	AES_CM, NULL	AES_CM
SRTP and SRTCP auth transf.	HMAC-SHA1	HMAC-SHA1
SRTP and SRTCP auth params:		
n_tag (tag length)	80	80
SRTP prefix_length	0	0
Key derivation PRF	AES_CM	AES_CM
Key material params (for each master key):		

```
  master key length                 128               128
  n_e (encr session key length)     128               128
  n_a (auth session key length)     160               160
  master salt key
  length of the master salt         112               112
  n_s (session salt key length)     112               112
  key derivation rate                 0                 0

  key lifetime
     SRTP-packets-max-lifetime      2^48              2^48
     SRTCP-packets-max-lifetime     2^31              2^31
     from-to-lifetime <"From, "To">
  MKI indicator                       0                 0
  length of the MKI                   0                 0
  value of the MKI

Crypto context index params:
  SSRC value
  ROC
  SEQ
  SRTCP Index
  Transport address
  Port number

Relation to other RTP profiles:
  sender's order between FEC and SRTP FEC-SRTP      FEC-SRTP
  (see Section 10)
```

9. Security Considerations

9.1 SSRC collision and two-time pad

Any fixed keystream output, generated from the same key and index MUST only be used to encrypt once. Re-using such keystream (jokingly called a "two-time pad" system by cryptographers), can seriously compromise security. The NSA's VENONA project [C99] provides a historical example of such a compromise. It is REQUIRED that automatic key management be used for establishing and maintaining SRTP and SRTCP keying material; this requirement is to avoid keystream reuse, which is more likely to occur with manual key management. Furthermore, in SRTP, a "two-time pad" is avoided by requiring the key, or some other parameter of cryptographic significance, to be unique per RTP/RTCP stream and packet. The pre-defined SRTP transforms accomplish packet-uniqueness by including the packet index and stream-uniqueness by inclusion of the SSRC.

The pre-defined transforms (AES-CM and AES-f8) allow master keys to be shared across streams belonging to the same RTP session by the inclusion of the SSRC in the IV. A master key MUST NOT be shared among different RTP sessions.

Thus, the SSRC MUST be unique between all the RTP streams within the same RTP session that share the same master key. RTP itself provides an algorithm for detecting SSRC collisions within the same RTP session. Thus, temporary collisions could lead to temporary two-time pad, in the unfortunate event that SSRCs collide at a point in time when the streams also have identical sequence numbers (occurring with probability roughly 2^(-48)). Therefore, the key management SHOULD take care of avoiding such SSRC collisions by including the SSRCs to be used in the session as negotiation parameters, proactively assuring their uniqueness. This is a strong requirements in scenarios where for example, there are multiple senders that can start to transmit simultaneously, before SSRC collision are detected at the RTP level.

Note also that even with distinct SSRCs, extensive use of the same key might improve chances of probabilistic collision and time-memory-tradeoff attacks succeeding.

As described, master keys MAY be shared between streams belonging to the same RTP session, but it is RECOMMENDED that each SSRC have its own master key. When master keys are shared among SSRC participants and SSRCs are managed by a key management module as recommended above, the RECOMMENDED policy for an SSRC collision error is for the participant to leave the SRTP session as it is a sign of malfunction.

9.2 Key Usage

The effective key size is determined (upper bounded) by the size of the master key and, for encryption, the size of the salting key. Any additive stream cipher is vulnerable to attacks that use statistical knowledge about the plaintext source to enable key collision and time-memory tradeoff attacks [MF00,H80,BS00]. These attacks take advantage of commonalities among plaintexts, and provide a way for a cryptanalyst to amortize the computational effort of decryption over many keys, or over many bytes of output, thus reducing the effective key size of the cipher. A detailed analysis of these attacks and their applicability to the encryption of Internet traffic is provided in [MF00]. In summary, the effective key size of SRTP when used in a security system in which m distinct keys are used, is equal to the key size of the cipher less the logarithm (base two) of m. Protection against such attacks can be provided simply by increasing the size of the keys used, which

here can be accomplished by the use of the salting key. Note that the salting key MUST be random but MAY be public. A salt size of (the suggested) size 112 bits protects against attacks in scenarios where at most 2^112 keys are in use. This is sufficient for all practical purposes.

Implementations SHOULD use keys that are as large as possible. Please note that in many cases increasing the key size of a cipher does not affect the throughput of that cipher.

The use of the SRTP and SRTCP indexes in the pre-defined transforms fixes the maximum number of packets that can be secured with the same key. This limit is fixed to 2^48 SRTP packets for an SRTP stream, and 2^31 SRTCP packets, when SRTP and SRTCP are considered independently. Due to for example re-keying, reaching this limit may or may not coincide with wrapping of the indices, and thus the sender MUST keep packet counts. However, when the session keys for related SRTP and SRTCP streams are derived from the same master key (the default behavior, Section 4.3), the upper bound that has to be considered is in practice the minimum of the two quantities. That is, when 2^48 SRTP packets or 2^31 SRTCP packets have been secured with the same key (whichever occurs before), the key management MUST be called to provide new master key(s) (previously stored and used keys MUST NOT be used again), or the session MUST be terminated. If a sender of RTCP discovers that the sender of SRTP (or SRTCP) has not updated the master or session key prior to sending 2^48 SRTP (or 2^31 SRTCP) packets belonging to the same SRTP (SRTCP) stream, it is up to the security policy of the RTCP sender how to behave, e.g. whether an RTCP BYE-packet should be sent and/or if the event should be logged.

Note: in most typical applications (assuming at least one RTCP packet for every 128,000 RTP packets), it will be the SRTCP index that first reaches the upper limit, although the time until this occurs is very long: even at 200 SRTCP packets/sec, the 2^31 index space of SRTCP is enough to secure approximately 4 months of communication.

Note that if the master key is to be shared between SRTP streams within the same RTP session (Section 9.1), although the above bounds are on a per stream (i.e. per SSRC) basis, the sender MUST base re-key decision on the stream whose sequence number space is the first to be exhausted.

Key derivation limits the amount of plaintext that is encrypted with a fixed session key, and made available to an attacker for analysis, but key derivation does not extend the master key's lifetime. To see this, simply consider our requirements to avoid two-time pad:

two distinct packets MUST either be processed with distinct IVs, or with distinct session keys, and both the distinctness of IV and of the session keys are (for the pre-defined transforms) dependent on the distinctness of the packet indices.

Note that with the key derivation, the effective key size is at most that of the master key, even if the derived session key is considerably longer. With the pre-defined authentication transform, the session authentication key is 160 bits, but the master key by default is only 128 bits. This design choice was made to comply with certain recommendations in [RFC2104] so that an existing HMAC implementation can be plugged into SRTP without problems. Since the default tag size is 80 bits, it is, for the applications in mind, also considered acceptable from security point of view. Users having concerns about this are RECOMMENDED to instead use a 192 bit master key in the key derivation. It was, however, chosen not to mandate 192-bit keys since existing AES implementations to be used in the key-derivation may not always support key-lengths other than 128 bits. Since AES is not defined (or properly analyzed) for use with 160 bit keys it is NOT RECOMMENDED that ad-hoc key-padding schemes are used to pad shorter keys to 192 or 256 bits.

9.3 Confidentiality of the RTP Payload

SRTP's pre-defined ciphers are "seekable" stream ciphers, i.e. ciphers able to efficiently seek to arbitrary locations in their keystream (so that the encryption or decryption of one packet does not depend on preceding packets). By using seekable stream ciphers, SRTP avoids the denial of service attacks that are possible on stream ciphers that lack this property. It is important to be aware that, as with any stream cipher, the exact length of the payload is revealed by the encryption. This means that it may be possible to deduce certain "formatting bits" of the payload, as the length of the codec output might vary due to certain parameter settings etc. This, in turn, implies that the corresponding bit of the keystream can be deduced. However, if the stream cipher is secure (counter mode and f8 are provably secure under certain assumptions [BDJR,KSYH]), knowledge of a few bits of the keystream will not aid an attacker in predicting subsequent keystream bits. Thus, the payload length (and information deducible from this) will leak, but nothing else.

As some RTP packet could contain highly predictable data, e.g. SID, it is important to use a cipher designed to resist known plaintext attacks (which is the current practice).

9.4 Confidentiality of the RTP Header

In SRTP, RTP headers are sent in the clear to allow for header compression. This means that data such as payload type, synchronization source identifier, and timestamp are available to an eavesdropper. Moreover, since RTP allows for future extensions of headers, we cannot foresee what kind of possibly sensitive information might also be "leaked".

SRTP is a low-cost method, which allows header compression to reduce bandwidth. It is up to the endpoints' policies to decide about the security protocol to employ. If one really needs to protect headers, and is allowed to do so by the surrounding environment, then one should also look at alternatives, e.g., IPsec [RFC2401].

9.5 Integrity of the RTP payload and header

SRTP messages are subject to attacks on their integrity and source identification, and these risks are discussed in Section 9.5.1. To protect against these attacks, each SRTP stream SHOULD be protected by HMAC-SHA1 [RFC2104] with an 80-bit output tag and a 160-bit key, or a message authentication code with equivalent strength. Secure RTP SHOULD NOT be used without message authentication, except under the circumstances described in this section. It is important to note that encryption algorithms, including AES Counter Mode and f8, do not provide message authentication. SRTCP MUST NOT be used with weak (or NULL) authentication.

SRTP MAY be used with weak authentication (e.g. a 32-bit authentication tag), or with no authentication (the NULL authentication algorithm). These options allow SRTP to be used to provide confidentiality in situations where

* weak or null authentication is an acceptable security risk, and
* it is impractical to provide strong message authentication.

These conditions are described below and in Section 7.5. Note that both conditions MUST hold in order for weak or null authentication to be used. The risks associated with exercising the weak or null authentication options need to be considered by a security audit prior to their use for a particular application or environment given the risks, which are discussed in Section 9.5.1.

Weak authentication is acceptable when the RTP application is such that the effect of a small fraction of successful forgeries is negligible. If the application is stateless, then the effect of a single forged RTP packet is limited to the decoding of that particular packet. Under this condition, the size of the

authentication tag MUST ensure that only a negligible fraction of the packets passed to the RTP application by the SRTP receiver can be forgeries. This fraction is negligible when an adversary, if given control of the forged packets, is not able to make a significant impact on the output of the RTP application (see the example of Section 7.5).

Weak or null authentication MAY be acceptable when it is unlikely that an adversary can modify ciphertext so that it decrypts to an intelligible value. One important case is when it is difficult for an adversary to acquire the RTP plaintext data, since for many codecs, an adversary that does not know the input signal cannot manipulate the output signal in a controlled way. In many cases it may be difficult for the adversary to determine the actual value of the plaintext. For example, a hidden snooping device might be required in order to know a live audio or video signal. The adversary's signal must have a quality equivalent to or greater than that of the signal under attack, since otherwise the adversary would not have enough information to encode that signal with the codec used by the victim. Plaintext prediction may also be especially difficult for an interactive application such as a telephone call.

Weak or null authentication MUST NOT be used when the RTP application makes data forwarding or access control decisions based on the RTP data. In such a case, an attacker may be able to subvert confidentiality by causing the receiver to forward data to an attacker. See Section 3 of [B96] for a real-life example of such attacks.

Null authentication MUST NOT be used when a replay attack, in which an adversary stores packets then replays them later in the session, could have a non-negligible impact on the receiver. An example of a successful replay attack is the storing of the output of a surveillance camera for a period of time, later followed by the injection of that output to the monitoring station to avoid surveillance. Encryption does not protect against this attack, and non-null authentication is REQUIRED in order to defeat it.

If existential message forgery is an issue, i.e. when the accuracy of the received data is of non-negligible importance, null authentication MUST NOT be used.

9.5.1. Risks of Weak or Null Message Authentication

During a security audit considering the use of weak or null authentication, it is important to keep in mind the following attacks which are possible when no message authentication algorithm is used.

An attacker who cannot predict the plaintext is still always able to modify the message sent between the sender and the receiver so that it decrypts to a random plaintext value, or to send a stream of bogus packets to the receiver that will decrypt to random plaintext values. This attack is essentially a denial of service attack, though in the absence of message authentication, the RTP application will have inputs that are bit-wise correlated with the true value. Some multimedia codecs and common operating systems will crash when such data are accepted as valid video data. This denial of service attack may be a much larger threat than that due to an attacker dropping, delaying, or re-ordering packets.

An attacker who cannot predict the plaintext can still replay a previous message with certainty that the receiver will accept it. Applications with stateless codecs might be robust against this type of attack, but for other, more complex applications these attacks may be far more grave.

An attacker who can predict the plaintext can modify the ciphertext so that it will decrypt to any value of her choosing. With an additive stream cipher, an attacker will always be able to change individual bits.

An attacker may be able to subvert confidentiality due to the lack of authentication when a data forwarding or access control decision is made on decrypted but unauthenticated plaintext. This is because the receiver may be fooled into forwarding data to an attacker, leading to an indirect breach of confidentiality (see Section 3 of [B96]). This is because data-forwarding decisions are made on the decrypted plaintext; information in the plaintext will determine to what subnet (or process) the plaintext is forwarded in ESP [RFC2401] tunnel mode (respectively, transport mode). When Secure RTP is used without message authentication, it should be verified that the application does not make data forwarding or access control decisions based on the decrypted plaintext.

Some cipher modes of operation that require padding, e.g. standard cipher block chaining (CBC) are very sensitive to attacks on confidentiality if certain padding types are used in the absence of integrity. The attack [V02] shows that this is indeed the case for the standard RTP padding as discussed in reference to Figure 1, when used together with CBC mode. Later transform additions to SRTP MUST therefore carefully consider the risk of using this padding without proper integrity protection.

9.5.2 Implicit Header Authentication

The IV formation of the f8-mode gives implicit authentication (IHA) of the RTP header, even when message authentication is not used. When IHA is used, an attacker that modifies the value of the RTP header will cause the decryption process at the receiver to produce random plaintext values. While this protection is not equivalent to message authentication, it may be useful for some applications.

10. Interaction with Forward Error Correction mechanisms

The default processing when using Forward Error Correction (e.g. RFC 2733) processing with SRTP SHALL be to perform FEC processing prior to SRTP processing on the sender side and to perform SRTP processing prior to FEC processing on the receiver side. Any change to this ordering (reversing it, or, placing FEC between SRTP encryption and SRTP authentication) SHALL be signaled out of band.

11. Scenarios

SRTP can be used as security protocol for the RTP/RTCP traffic in many different scenarios. SRTP has a number of configuration options, in particular regarding key usage, and can have impact on the total performance of the application according to the way it is used. Hence, the use of SRTP is dependent on the kind of scenario and application it is used with. In the following, we briefly illustrate some use cases for SRTP, and give some guidelines for recommended setting of its options.

11.1 Unicast

A typical example would be a voice call or video-on-demand application.

Consider one bi-directional RTP stream, as one RTP session. It is possible for the two parties to share the same master key in the two directions according to the principles of Section 9.1. The first round of the key derivation splits the master key into any or all of the following session keys (according to the provided security functions):

SRTP_encr_key, SRTP_auth_key, SRTCP_encr_key, and SRTCP_auth key.

(For simplicity, we omit discussion of the salts, which are also derived.) In this scenario, it will in most cases suffice to have a single master key with the default lifetime. This guarantees sufficiently long lifetime of the keys and a minimum set of keys in place for most practical purposes. Also, in this case RTCP

protection can be applied smoothly. Under these assumptions, use of the MKI can be omitted. As the key-derivation in combination with large difference in the packet rate in the respective directions may require simultaneous storage of several session keys, if storage is an issue, we recommended to use low-rate key derivation.

The same considerations can be extended to the unicast scenario with multiple RTP sessions, where each session would have a distinct master key.

11.2 Multicast (one sender)

Just as with (unprotected) RTP, a scalability issue arises in big groups due to the possibly very large amount of SRTCP Receiver Reports that the sender might need to process. In SRTP, the sender may have to keep state (the cryptographic context) for each receiver, or more precisely, for the SRTCP used to protect Receiver Reports. The overhead increases proportionally to the size of the group. In particular, re-keying requires special concern, see below.

Consider first a small group of receivers. There are a few possible setups with the distribution of master keys among the receivers. Given a single RTP session, one possibility is that the receivers share the same master key as per Section 9.1 to secure all their respective RTCP traffic. This shared master key could then be the same one used by the sender to protect its outbound SRTP traffic. Alternatively, it could be a master key shared only among the receivers and used solely for their SRTCP traffic. Both alternatives requires the receivers to trust each other.

Considering SRTCP and key storage, it is recommended to use low-rate (or zero) key_derivation (except the mandatory initial one), so that the sender does not need to store too many session keys (each SRTCP stream might otherwise have a different session key at a given point in time, as the SRTCP sources send at different times). Thus, in case key derivation is wanted for SRTP, the cryptographic context for SRTP can be kept separate from the SRTCP crypto context, so that it is possible to have a key_derivation_rate of 0 for SRTCP and a non-zero value for SRTP.

Use of the MKI for re-keying is RECOMMENDED for most applications (see Section 8.1).

If there are more than one SRTP/SRTCP stream (within the same RTP session) that share the master key, the upper limit of 2^48 SRTP packets / 2^31 SRTCP packets means that, before one of the streams reaches its maximum number of packets, re-keying MUST be triggered

on ALL streams sharing the master key. (From strict security point of view, only the stream reaching the maximum would need to be re-keyed, but then the streams would no longer be sharing master key, which is the intention.) A local policy at the sender side should force rekeying in a way that the maximum packet limit is not reached on any of the streams. Use of the MKI for re-keying is RECOMMENDED.

In large multicast with one sender, the same considerations as for the small group multicast hold. The biggest issue in this scenario is the additional load placed at the sender side, due to the state (cryptographic contexts) that has to be maintained for each receiver, sending back RTCP Receiver Reports. At minimum, a replay window might need to be maintained for each RTCP source.

11.3 Re-keying and access control

Re-keying may occur due to access control (e.g., when a member is removed during a multicast RTP session), or for pure cryptographic reasons (e.g. the key is at the end of its lifetime). When using SRTP default transforms, the master key MUST be replaced before any of the index spaces are exhausted for any of the streams protected by one and the same master key.

How key management re-keys SRTP implementations is out of scope, but it is clear that there are straightforward ways to manage keys for a multicast group. In one-sender multicast, for example, it is typically the responsibility of the sender to determine when a new key is needed. The sender is the one entity that can keep track of when the maximum number of packets has been sent, as receivers may join and leave the session at any time, there may be packet loss and delay etc. In scenarios other than one-sender multicast, other methods can be used. Here, one must take into consideration that key exchange can be a costly operation, taking several seconds for a single exchange. Hence, some time before the master key is exhausted/expires, out-of-band key management is initiated, resulting in a new master key that is shared with the receiver(s). In any event, to maintain synchronization when switching to the new key, group policy might choose between using the MKI and the <"From", "To">, as described in Section 8.1.

For access control purposes, the <"From", "To"> periods are set at the desired granularity, dependent on the packet rate. High rate re-keying can be problematic for SRTCP in some large-group scenarios. As mentioned, there are potential problems in using the SRTP index, rather than the SRTCP index, for determining the master key. In particular, for short periods during switching of master keys, it may be the case that SRTCP packets are not under the

current master key of the correspondent SRTP. Therefore, using the MKI for re-keying in such scenarios will produce better results.

11.4 Summary of basic scenarios

The description of these scenarios highlights some recommendations on the use of SRTP, mainly related to re-keying and large scale multicast:

- Do not use fast re-keying with the <"From", "To"> feature. It may, in particular, give problems in retrieving the correct SRTCP key, if an SRTCP packet arrives close to the re-keying time. The MKI SHOULD be used in this case.

- If multiple SRTP streams in the same RTP session share the same master key, also moderate rate re-keying MAY have the same problems, and the MKI SHOULD be used.

- Though offering increased security, a non-zero key_derivation_rate is NOT RECOMMENDED when trying to minimize the number of keys in use with multiple streams.

12. IANA Considerations

The RTP specification establishes a registry of profile names for use by higher-level control protocols, such as the Session Description Protocol (SDP), to refer to transport methods. This profile registers the name "RTP/SAVP".

SRTP uses cryptographic transforms, which a key management protocol signals. It is the task of each particular key management protocol to register the cryptographic transforms or suites of transforms with IANA. The key management protocol conveys these protocol numbers, not SRTP, and each key management protocol chooses the numbering scheme and syntax that it requires.

Specification of a key management protocol for SRTP is out of scope here. Section 8.2, however, provides guidance on the parameters that need to be defined for the default and mandatory transforms.

13. Acknowledgements

David Oran (Cisco) and Rolf Blom (Ericsson) are co-authors of this document but their valuable contributions are acknowledged here to keep the length of the author list down.

The authors would in addition like to thank Magnus Westerlund, Brian Weis, Ghyslain Pelletier, Morgan Lindqvist, Robert Fairlie-

Cuninghame, Adrian Perrig, the AVT WG and in particular the chairmen Colin Perkins and Stephen Casner, the Transport and Security Area Directors, and Eric Rescorla for their reviews and support.

14. Author's Addresses

Questions and comments should be directed to the authors and avt@ietf.org:

Mark Baugher
Cisco Systems, Inc.
5510 SW Orchid Street Phone: +1 408-853-4418
Portland, OR 97219 USA Email: mbaugher@cisco.com

Elisabetta Carrara
Ericsson Research
SE-16480 Stockholm Phone: +46 8 50877040
Sweden EMail: elisabetta.carrara@ericsson.com

David A. McGrew
Cisco Systems, Inc.
San Jose, CA 95134-1706 Phone: +1 301-349-5815
USA EMail: mcgrew@cisco.com

Mats Naslund
Ericsson Research
SE-16480 Stockholm Phone: +46 8 58533739
Sweden EMail: mats.naslund@ericsson.com

Karl Norrman
Ericsson Research
SE-16480 Stockholm Phone: +46 8 4044502
Sweden EMail: karl.norrman@ericsson.com

15. References

Normative

[AES] NIST, "Advanced Encryption Standard (AES)", FIPS PUB 197, http://www.nist.gov/aes/

[AVPNEW] Schulzrinne, H., Casner, S., RTP Profile for Audio and Video Conferences with Minimal Control, IETF Work in Progress, March 2003.

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R.: "HMAC: Keyed-hashing for message authentication". IETF RFC 2104, February 1997.

[RTPNEW] Schulzrinne, H., Casner, S., Frederick, R., Jacobson,V., "RTP: A Transport Protocol for Real-time Applications", IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-new-12.txt, March 2003.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997.

[RFC2401] Kent, S., and R. Atkinson, "Security Architecture for IP", IETF RFC 2401, November 1998.

[RFC2675] Borman, D., Deering, S., Hinden, R., "IPv6 Jumbograms", IETF RFC 2675, August 1999.

[RFC2828] Shirey, R., "Internet Security Glossary", IETF RFC 2828, May 2000.

Informative

[AES-CTR] Lipmaa, H., Rogaway, P., Wagner, D., "CTR-Mode Encryption", NIST, http://csrc.nist.gov/encryption/modes/workshop1/papers/lipmaa-ctr.pdf

[B96] Bellovin, S., "Problem Areas for the IP Security Protocols," in Proceedings of the Sixth Usenix Unix Security Symposium, pp. 1-16, San Jose, CA, July 1996 (http://www.research.att.com/~smb/papers/index.html).

[BDJR] Bellare, M., Desai, A., Jokipii, E., and Rogaway, P., "A Concrete Treatment of Symmetric Encryption: Analysis of DES Modes of Operation", Proceedings 38th IEEE FOCS, pp. 394-403, 1997.

[BS00] Biryukov, A. and Shamir, A., "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers", Proceedings, ASIACRYPT 2000, LNCS 1976, pp. 1-13, Springer Verlag.

[C99] Crowell, W. P., "Introduction to the VENONA Project", http://www.nsa.gov:8080/docs/venona/index.html.

[CTR] Dworkin, M., NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[f8-a] 3GPP TS 35.201 V4.1.0 (2001-12) Technical Specification 3rd Generation Partnership Project; Technical Specification Group

Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification(Release 4).

[f8-b] 3GPP TR 33.908 V4.0.0 (2001-09) Technical Report 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; General Report on the Design, Specification and Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms (Release 4).

[GDOI] Baugher, M., Hardjono, T., Harney, H., and Weis, B., "The Group Domain of Interpretation, Accepted as IETF Proposed Standard, http://www.ietf.org/internet-drafts/draft-ietf-msec-gdoi-07.txt, 2003

[HAC] Menezes, A., Van Oorschot, P., and Vanstone, S., "Handbook of Applied Cryptography", CRC Press, 1997, ISBN 0-8493-8523-7.

[H80] Hellman, M. E., "A cryptanalytic time-memory trade-off", IEEE Transactions on Information Theory, July 1980, pp. 401-406.

[KINK] Thomas, M., Vilhuber, J., "Kerberized Internet Negotiation of Keys (KINK) ", IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-kink-kink-05.txt, January 2003

[KEYMGT] Arrko, J. et. al. Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP), IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-kmgmt-ext-07.txt, February 2003

[KSYH] Kang, J-S., Shin, S-U., Hong, D., and Yi, O., "Provable Security of KASUMI and 3GPP Encryption Mode f8", Proceedings Asiacrypt 2001, Springer Verlag LNCS 2248, pp. 255-271, 2001.

[MIKEY] Arrko, J., et. al., "MIKEY: Multimedia Internet KEYing", IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-msec-mikey-06.txt, February 2003.

[MF00] McGrew, D., and Fluhrer, S., "Attacks on Encryption of Redundant Plaintext and Implications on Internet Security", the Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography (SAC 2000), Springer-Verlag.

[RK99] Rescorla, E., and Korver, B., "Guidelines for Writing RFC Text on Security Considerations," draft-rescorla-sec-cons-00.txt

[PCST1] Perrig, A., Canetti, R., Tygar, D., Song, D., "Efficient and Secure Source Authentication for Multicast", in Proc. of Network and Distributed System Security Symposium NDSS 2001, pp. 35-46, 2001.

[PCST2] Perrig, A., Canetti, R., Tygar, D., Song, D., "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", in Proc. of IEEE Security and Privacy Symposium S&P2000, pp. 56-73, 2000.

[RFC3095] Bormann et al., "RObust Header Compression: Framework and four profiles: RTP, UDP, ESP, and uncompressed (ROHC)", RFC 3095, July 2001.

[RFC3242] Jonsson, L-E., Pelletier, G., "RObust Header Compression (ROHC): A Link-Layer Assisted Profile for IP/UDP/RTP ", IETF RFC 3242, April 2002.

[SDMS] Baugher, M., Wing, D., "SDP Security Descriptions for Media Streams," IETF, Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdescriptions-00.txt, February 2003.

[SWO] Svanbro, K., Wiorek, J., and Olin, B., "Voice-over-IP-over-wireless", Proc. PIMRC 2000, London, Sept. 2000.

[V02] Vaudenay, S., "Security Flaws Induced by CBC Padding - Application to SSL, IPsec, WTLS...", Advances in Cryptology, EUROCRYPT'02, LNCS 2332, pp. 534-545.

[WC81] Wegman, M. N., and Carter, J.L, "New Hash Functions and Their Use in Authentication and Set Equality", JCSS 22, 265-279, 1981.

16. Intellectual Property Right Considerations

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such

proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

17. Full Copyright Statement

Appendix A: Pseudocode for Index Determination

The following is an example of pseudo-code for the algorithm to determine the index i of an SRTP packet with sequence number SEQ. In the following, signed arithmetic is assumed.

```
if (s_l < 32,768)
   if (SEQ - s_l > 32,768)
      set v to (ROC-1) mod 2^32
   else
      set v to ROC
   endif
else
   if (s_l - 32,768 > SEQ)
      set v to (ROC+1) mod 2^32
   else
      set v to ROC
   endif
endif
return SEQ + v*65,536
```

Appendix B: Test Vectors

All values are in hexadecimal.

B.1 AES-f8 Test Vectors

```
SRTP PREFIX LENGTH  :   0

RTP packet header   :   806e5cba50681de55c621599

RTP packet payload  :   70736575646f72616e646f6d6e657373
                        206973207468652065787420626573
                        74207468696e67

ROC                 :   d462564a
key                 :   234829008467be186c3de14aae72d62c
salt key            :   32f2870d
key-mask (m)        :   32f2870d55555555555555555555555
key XOR key-mask    :   11baae0dd132eb4d3968b41ffb278379

IV                  :   006e5cba50681de55c621599d462564a
IV'                 :   595b699bbd3bc0df26062093c1ad8f73

j                   :   0
IV' XOR j           :   595b699bbd3bc0df26062093c1ad8f73
```

```
S(-1)               :   00000000000000000000000000000000
S(-1) XOR IV' XOR j :   595b699bbd3bc0df26062093c1ad8f73
S(0)                :   71ef82d70a172660240709c7fbb19d8e
plaintext           :   70736575646f72616e646f6d6e657373
ciphertext          :   019ce7a26e7854014a6366aa95d4eefd

j                   :   1
IV' XOR j           :   595b699bbd3bc0df26062093c1ad8f72
S(0)                :   71ef82d70a172660240709c7fbb19d8e
S(0) XOR IV' XOR j  :   28b4eb4cb72ce6bf020129543a1c12fc
S(1)                :   3abd640a60919fd43bd289a09649b5fc
plaintext           :   20697320746865206e65787420626573
ciphertext          :   1ad4172a14f9faf455b7f1d4b62bd08f

j                   :   2
IV' XOR j           :   595b699bbd3bc0df26062093c1ad8f70
S(1)                :   3abd640a60919fd43bd289a09649b5fc
S(1) XOR IV' XOR j  :   63e60d91ddaa5f0b1dd4a93357e43a8c
S(2)                :   584d14a591acfca846b3aa3a0ab50fec
plaintext           :   74207468696e67
ciphertext          :   2c6d60cdf8c29b
```

B.2 AES-CM Test Vectors

```
Keystream segment length: 1044512 octets (65282 AES blocks)
Session Key:     2B7E151628AED2A6ABF7158809CF4F3C
Rollover Counter: 00000000
Sequence Number:  0000
SSRC:             00000000
Session Salt:     F0F1F2F3F4F5F6F7F8F9FAFBFCFD0000 (already shifted)
Offset:           F0F1F2F3F4F5F6F7F8F9FAFBFCFD0000

Counter                           Keystream

F0F1F2F3F4F5F6F7F8F9FAFBFCFD0000  E03EAD0935C95E80E166B16DD92B4EB4
F0F1F2F3F4F5F6F7F8F9FAFBFCFD0001  D23513162B02D0F72A43A2FE4A5F97AB
F0F1F2F3F4F5F6F7F8F9FAFBFCFD0002  41E95B3BB0A2E8DD477901E4FCA894C0
...                               ...
F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF  EC8CDF7398607CB0F2D21675EA9EA1E4
F0F1F2F3F4F5F6F7F8F9FAFBFCFDFF00  362B7C3C6773516318A077D7FC5073AE
F0F1F2F3F4F5F6F7F8F9FAFBFCFDFF01  6A2CC3787889374FBEB4C81B17BA6C44
```

Nota Bene: this test case is contrived so that the latter part of the keystream segment coincides with the test case in Section F.5.1 of [CTR].

B.3 Key Derivation Test Vectors

This section provides test data for the default key derivation function, which uses AES-128 in Counter Mode. In the following, we walk through the initial key derivation for the AES-128 Counter Mode cipher, which requires a 16 octet session encryption key and a 14 octet session salt, and an authentication function which requires a 94-octet session authentication key. These values are called the cipher key, the cipher salt, and the auth key in the following. Since this is the initial key derivation, the value of (index DIV key_derivation_rate) is zero (actually, a six-octet string of zeros). In the following, we shorten key_derivation_rate to kdr.

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

```
master key:  E1F97A0D3E018BE0D64FA32C06DE4139
master salt: 0EC675AD498AFEEBB6960B3AABE6
```

We first show how the cipher key is generated. The input block for AES-CM is generated by exclusive-oring the master salt with the concatenation of the encryption key label 0x00 with (index DIV kdr), then padding on the right with two null octets (which implements the multiply-by-2^16 operation, see Section 4.3.3). The resulting value is then AES-CM- encrypted using the master key to get the cipher key.

```
index DIV kdr:                000000000000
label:                      00
master salt:   0EC675AD498AFEEBB6960B3AABE6
-----------------------------------------------
xor:           0EC675AD498AFEEBB6960B3AABE6     (x, PRF input)

x*2^16:        0EC675AD498AFEEBB6960B3AABE60000 (AES-CM input)

cipher key:    C61E7A93744F39EE10734AFE3FF7A087 (AES-CM output)
```

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

```
index DIV kdr:                000000000000
label:                      02
master salt:   0EC675AD498AFEEBB6960B3AABE6

-----------------------------------------------
xor:           0EC675AD498AFEE9B6960B3AABE6     (x, PRF input)
```

```
   x*2^16:        0EC675AD498AFEE9B6960B3AABE60000 (AES-CM input)

                  30CBBC08863D8C85D49DB34A9AE17AC6 (AES-CM ouptut)

   cipher salt:   30CBBC08863D8C85D49DB34A9AE1
```

We now show how the auth key is generated. The input block for AES-CM is generated as above, but using the authentication key label.

```
   index DIV kdr:                   000000000000
   label:                         01
   master salt:     0EC675AD498AFEEBB6960B3AABE6
   ------------------------------------------------
   xor:             0EC675AD498AFEEAB6960B3AABE6     (x, PRF input)

   x*2^16:          0EC675AD498AFEEAB6960B3AABE60000 (AES-CM input)
```

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

```
auth key                          AES input blocks
CEBE321F6FF7716B6FD4AB49AF256A15  0EC675AD498AFEEAB6960B3AABE60000
6D38BAA48F0A0ACF3C34E2359E6CDBCE  0EC675AD498AFEEAB6960B3AABE60001
E049646C43D9327AD175578EF7227098  0EC675AD498AFEEAB6960B3AABE60002
6371C10C9A369AC2F94A8C5FBCDDDC25  0EC675AD498AFEEAB6960B3AABE60003
6D6E919A48B610EF17C2041E47403576  0EC675AD498AFEEAB6960B3AABE60004
6B68642C59BBFC2F34DB60DBDFB2      0EC675AD498AFEEAB6960B3AABE60005
```

This draft expires in December 2003.