Network Working Group
Request for Comments: 4568
Category: Standards Track

F. Andreasen
M. Baugher
D. Wing
Cisco Systems

July 2006

Session Description Protocol (SDP) Security Descriptions for Media Streams

Status of This Memo

Copyright Notice

Abstract

This document defines a Session Description Protocol (SDP) cryptographic attribute for unicast media streams. The attribute describes a cryptographic key and other parameters that serve to configure security for a unicast media stream in either a single message or a roundtrip exchange. The attribute can be used with a variety of SDP media transports, and this document defines how to use it for the Secure Real-time Transport Protocol (SRTP) unicast media streams. The SDP crypto attribute requires the services of a data security protocol to secure the SDP message.

Table of Contents

1. Introduction

The Session Description Protocol (SDP) [RFC4566] describes multimedia sessions, which can be audio, video, whiteboard, fax, modem, and other media streams. Security services such as data origin authentication, integrity, and confidentiality are often needed for those streams. The Secure Real-time Transport Protocol (SRTP) [RFC3711] provides security services for RTP media and is signaled by use of secure RTP transport (e.g., "RTP/SAVP" or "RTP/SAVPF") in an SDP media (m=) line. However, there are no means within SDP itself to configure SRTP beyond using default values. This document specifies a new SDP attribute called "crypto", which is used to signal and negotiate cryptographic parameters for media streams in general, and for SRTP in particular. The definition of the crypto attribute in this document is limited to two-party unicast media streams where each source has a unique cryptographic key; support for multicast media streams or multipoint unicast streams is for further study.

The crypto attribute is defined in a generic way to enable its use with SRTP and any other secure transports that can establish cryptographic parameters with only a single message or in a single round-trip exchange using the offer/answer model [RFC3264]. Extensions to transports other than SRTP, however, is beyond the scope of this document. Each type of secure media transport needs its own specification for the crypto-attribute parameter. These definitions are frequently unique to the particular type of transport and must be specified in a Standards-Track RFC and registered with IANA according to the procedures defined in Section 10. This document defines the security parameters and keying material for SRTP only.

It would be self-defeating not to secure cryptographic keys and other parameters at least as well as the data are secured. Data security protocols such as SRTP rely upon a separate key management system to securely establish encryption and/or authentication keys. Key management protocols provide authenticated key establishment (AKE) procedures to authenticate the identity of each endpoint and protect against man-in-the-middle, reflection/replay, connection hijacking, and some denial-of-service attacks [skeme]. Along with the key, an AKE protocol such as MIKEY [mikey], GDOI [GDOI], KINK [kink], IKE [ike], Secure Multiparts [s/mime, pgp/mime], or TLS [TLS] securely disseminates information describing both the key and the data-security session. AKE is needed because it is pointless to provide a key over a medium where an attacker can snoop the key, alter the definition of the key to render it useless, or change the parameters of the security session to gain unauthorized access to session-related information.

SDP, however, was not designed to provide AKE services, and the media security descriptions defined in this document do not add AKE services to SDP. This specification is no replacement for a key management protocol or for the conveyance of key management messages in SDP [keymgt]. The SDP security descriptions defined here are suitable for restricted cases only where IPsec, TLS, or some other encapsulating data-security protocol (e.g., SIP S/MIME) protects the SDP message. This document adds security descriptions to those encrypted and/or authenticated SDP messages through the new SDP "crypto" attribute, which provides the cryptographic parameters of a media stream.

The "crypto" attribute can be adapted to any media transport, but its precise definition is unique to a particular transport.

In Section 2, we provide notational conventions followed by an applicability statement for the crypto attribute in Section 3. In Section 4, we introduce the general SDP crypto attribute, and in Section 5, we define how it is used with and without the offer/answer model. In Section 6, we define the crypto attribute details needed for SRTP, and in Section 7, we define SRTP-specific use of the attribute with and without the offer/answer model. Section 8 recites security considerations, and Section 9 gives an Augmented-BNF grammar for the general crypto attribute as well as the SRTP-specific use of the crypto attribute. IANA considerations are provided in Section 10.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. The terminology in this document conforms to [RFC2828], "Internet Security Glossary".

n^r is exponentiation, where n is multiplied by itself r times; n and r are integers. 0..k is an integer range of all integers from 0 through k, inclusive.

The terms 'transport' and 'media transport' are used to mean 'transport protocol' as defined in RFC 4566.

Explanatory notes are provided in several places throughout the document; these notes are indented three spaces from the surrounding text.

3. Applicability

RFC 4567 provides similar cryptographic key distribution capabilities and is intended for use when the signaling is to be confidential and/or integrity-protected separately from the keying material.

In contrast, this specification carries the keying material within the SDP message, and it is intended for use when the keying material is protected along with the signaling. Implementations MUST employ security mechanisms that provide confidentiality and integrity for the keying material. When this specification is used in the context of SIP [RFC3261], the application SHOULD employ either the SIPS URI or S/MIME to provide protection for the SDP message and the keying material that it contains. The use of transport layer or IP layer security in lieu of the SIPS URI or S/MIME protection is NOT RECOMMENDED since the protection of the SDP message and the keying material that it contains cannot be ensured through all intermediate entities such as SIP proxies.

4. SDP "Crypto" Attribute and Parameters

A new media-level SDP attribute called "crypto" describes the cryptographic suite, key parameters, and session parameters for the preceding unicast media line. The "crypto" attribute MUST only appear at the SDP media level (not at the session level). The "crypto" attribute follows the format (see Section 9.1 for the formal ABNF grammar):

```
a=crypto:<tag> <crypto-suite> <key-params> [<session-params>]
```

The fields tag, crypto-suite, key-params, and session-params are described in the following sub-sections. The values of each of these fields is case-insensitive, unless otherwise noted. However, implementers are encouraged to use the actual case shown in this document and any extensions to it. Note that per normal SDP rules, the "crypto" attribute name itself is case-sensitive. Below, we show an example of the crypto attribute for the "RTP/SAVP" transport, i.e., the secure RTP extension to the Audio/Video Profile [RFC3711]. In the following, newlines are included for formatting reasons only:

```
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:32
```

The crypto-suite is AES_CM_128_HMAC_SHA1_80, key-params is defined by the text starting with "inline:", and session-params is omitted.

4.1. Tag

The tag is a decimal number used as an identifier for a particular crypto attribute (see Section 9.1 for details); leading zeroes MUST NOT be used. The tag MUST be unique among all crypto attributes for a given media line. It is used with the offer/answer model to determine which of several offered crypto attributes were chosen by the answerer (see Section 5.1).

In the offer/answer model, the tag is a negotiated parameter.

4.2. Crypto-Suite

The crypto-suite field is an identifier that describes the encryption and authentication algorithms (e.g., AES_CM_128_HMAC_SHA1_80) for the transport in question (see Section 9.1 for details). The possible values for the crypto-suite parameter are defined within the context of the transport, i.e., each transport defines a separate namespace for the set of crypto-suites. For example, the crypto-suite "AES_CM_128_HMAC_SHA1_80" defined within the context "RTP/SAVP" transport applies to Secure RTP only; the string may be reused for another transport (e.g., "RTP/SAVPF" [srtpf]), but a separate definition would be needed.

In the offer/answer model, the crypto-suite is a negotiated parameter.

4.3. Key Parameters

The key-params field provides one or more sets of keying material for the crypto-suite in question. The field consists of a method indicator followed by a colon, and the actual keying information as shown below (the formal grammar is provided in Section 9.1):

```
key-params = <key-method> ":" <key-info>
```

Keying material might be provided by different means from that for key-params; however, this is out of scope. Only one method is defined in this document, namely, "inline", which indicates that the actual keying material is provided in the key-info field itself. There is a single name space for the key-method, i.e., the key-method is transport independent. New key-methods (e.g., use of a URL) may be defined in a Standards-Track RFC in the future. Although the key-method itself may be generic, the accompanying key-info definition is specific not only to the key-method, but also to the transport in question. Key-info encodes keying material for a crypto suite, which defines that keying material. New key methods MUST be registered with the IANA according to the procedures defined in Section 10.2.1.

Key-info is defined as a general octet string (see Section 9.1 for details); further transport and key-method specific syntax and semantics MUST be provided in a Standards-Track RFC for each combination of transport and key-method that uses it; definitions for SRTP are provided in Section 6. Note that such definitions are provided within the context of both a particular transport (e.g., "RTP/SAVP") and a specific key-method (e.g., "inline"). IANA will register the list of supported key methods for each transport.

When multiple keys are included in the key parameters, it MUST be possible to determine which of the keys is being used in a given media packet by a simple inspection of the media packet received; a trial-and-error approach between the possible keys MUST NOT be performed.

> For SRTP, this could be achieved by use of Master Key Identifiers (MKI) [RFC3711]. Use of <"From, "To"> values are not supported in SRTP security descriptions for reasons explained in Section 6.1, below.

In the offer/answer model, the key parameter is a declarative parameter.

4.4. Session Parameters

Session parameters are specific to a given transport and use of them is OPTIONAL in the security descriptions framework, where they are just defined as general character strings. If session parameters are to be used for a given transport, then transport-specific syntax and semantics MUST be provided in a Standards-Track RFC; definitions for SRTP are provided in Section 6.

In the offer/answer model, session parameters may be either negotiated or declarative; the definition of specific session parameters MUST indicate whether they are negotiated or declarative. Negotiated parameters apply to data sent in both directions, whereas declarative parameters apply only to media sent by the entity that generated the SDP. Thus, a declarative parameter in an offer applies to media sent by the offerer, whereas a declarative parameter in an answer applies to media sent by the answerer.

4.5. Example

This example shows use of the crypto attribute for the "RTP/SAVP" media transport type (as defined in Section 5). The "a=crypto" line is actually one long line; it is shown as two lines due to page formatting.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 161.44.17.12/127
t=2873397496 2873404696
m=video 51372 RTP/SAVP 31
a=crypto:1 AES_CM_128_HMAC_SHA1_80
 inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32
 inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
m=application 32416 udp wb
a=orient:portrait
```

This SDP message describes three media streams, two of which use the "RTP/SAVP" transport. Each has a crypto attribute for the "RTP/SAVP" transport. These secure-RTP specific descriptions are defined in Section 6.

5. General Use of the crypto Attribute

In this section, we describe the general use of the crypto attribute outside of any transport or key-method specific rules.

5.1. Use with Offer/Answer

The general offer/answer rules for the crypto attribute are in addition to the rules specified in RFC 3264, which MUST be followed, unless otherwise noted. RFC 3264 defines operation for both unicast and multicast streams; the sections below describe operation for two-party unicast streams only, since support for multicast streams (and multipoint unicast streams) is for further study.

5.1.1. Generating the Initial Offer - Unicast Streams

When generating an initial offer for a unicast stream, there MUST be one or more crypto attributes present for each media stream for which security is desired. Each crypto attribute for a given media stream MUST contain a unique tag.

The ordering of multiple "a=crypto" lines is significant: the most preferred crypto line is listed first. Each crypto attribute describes the crypto-suite, key(s), and possibly session parameters offered for the media stream. In general, a "more preferred" crypto-suite SHOULD be cryptographically stronger than a "less preferred" crypto-suite.

The crypto-suite always applies to media in the directions supported by the media stream (e.g., send and receive). The key(s), however, apply to data packets (e.g., SRTP and SRTCP packets) that will be sent by the same party that generated the SDP. That is, each endpoint determines its own transmission keys and sends those keys, in SDP, to the other endpoint.

> This is done for consistency. Also, in the case of SRTP, for example, secure RTCP will still be flowing in both the send and receive direction for a unidirectional stream.

The inline parameter conveys the keying material used by an endpoint to encrypt the media streams transmitted by that endpoint. The same keying material is used by the recipient to decrypt those streams.

The offer may include session parameters. There are no general offer rules for the session parameters; instead, specific rules may be provided as part of the transport-specific definitions of any session parameters.

When issuing an offer, the offerer MUST be prepared to support media security in accordance with any of the crypto attributes included in the offer. There are, however, two problems associated with this. First of all, the offerer does not know which key the answerer will be using for media sent to the offerer. Second, the offerer may not be able to deduce which of the offered crypto attributes were accepted. Since media may arrive prior to the answer, delay or clipping can occur. If this is unacceptable to the offerer, the offerer SHOULD use a mechanism outside the scope of this document to prevent the above problem.

For example, in SIP [RFC3261], a "security" precondition as defined in [sprecon] could solve the above problem.

5.1.2. Generating the Initial Answer - Unicast Streams

When the answerer receives the initial offer with one or more crypto attributes for a given unicast media stream, the answerer MUST either accept exactly one of the offered crypto attributes, or the offered stream MUST be rejected.

If the answerer wishes to indicate support for other crypto attributes, those can be listed by use of the SDP Simple Capability Declaration [RFC3407] extensions.

Only crypto attributes that are valid can be accepted; valid attributes do not violate any of the general rules defined for security descriptions, nor any specific rules defined for the transport and key-method in question. When selecting one of the valid crypto attributes, the answerer SHOULD select the most preferred crypto attribute it can support, i.e., the first valid supported crypto attribute in the list, according to the answerer’s capabilities and security policies.

If there are one or more crypto attributes in the offer, but none of them are valid or none of the valid ones are supported, the offered media stream MUST be rejected.

When an offered crypto attribute is accepted, the crypto attribute in the answer MUST contain the following:

* The tag and crypto-suite from the accepted crypto attribute in the offer (the same crypto-suite MUST be used in the send and receive direction).

* The key(s) the answerer will be using for media sent to the offerer. Note that a key MUST be provided, irrespective of any direction attributes in the offer or answer.

Furthermore, any session parameters that are negotiated MUST be included in the answer. Declarative session parameters provided by the offerer are not included in the answer; however, the answerer may provide its own set of declarative session parameters.

Once the answerer has accepted one of the offered crypto attributes, the answerer MAY begin sending media to the offerer in accordance with the selected crypto attribute. Note, however, that the offerer may not be able to process such media packets correctly until the answer has been received.

5.1.3. Processing of the Initial Answer - Unicast Streams

When the offerer receives the answer, the offerer MUST verify that one of the initially offered crypto suites and its accompanying tag were accepted and echoed in the answer. Also, the answer MUST include one or more keys, which will be used for media sent from the answerer to the offerer.

If the offer contained any mandatory negotiated session parameters (see Section 6.3.7), the offerer MUST verify that said parameters are included in the answer and support them. If the answer contains any mandatory declarative session parameters, the offerer MUST be able to support those.

If any of the above fails, the negotiation MUST fail.

5.1.4. Modifying the Session

Once a media stream has been established, it MAY be modified at any time, as described in RFC 3264, Section 8. Such a modification MAY be triggered by the security service, e.g., in order to perform a re-keying or change the crypto-suite. If media stream security using the general security descriptions defined here is still desired, the crypto attribute MUST be included in these new offer/answer exchanges. The procedures are similar to those defined in Section 5.1.1, 5.1.2, and 5.1.3 of this document, subject to the considerations provided in RFC 3264, Section 8.

5.2. Use Outside Offer/Answer

The crypto attribute can also be used outside the context of offer/answer where there is no negotiation of the crypto suite, cryptographic key, or session parameters. In this case, the sender determines security parameters for the stream. Since there is no negotiation mechanism, the sender MUST include exactly one crypto attribute, and the receiver MUST either accept it or SHOULD NOT

receive the associated stream. The sender SHOULD select the security description that it deems most secure for its purposes.

5.3. General Backwards Compatibility Considerations

In the offer/answer model, it is possible that the answerer supports a given secure transport (e.g., "RTP/SAVP") and accepts the offered media stream, but that the answerer does not support the crypto attribute defined in this document and hence ignores it. The offerer can recognize this situation by seeing an accepted media stream in the answer that does not include a crypto line. In that case, the security negotiation defined here MUST fail.

Similar issues exist when security descriptions are used outside the offer/answer model. But the source of a non-negotiated security description has no indication that the receiver has ignored the crypto attribute.

6. SRTP Security Descriptions

In this section, we provide definitions for security descriptions for SRTP media streams. In the next section, we define how to use SRTP security descriptions with and without the offer/answer model.

SRTP security descriptions MUST only be used with the SRTP transport (e.g., "RTP/SAVP" or "RTP/SAVPF"). The following specifies security descriptions for the "RTP/SAVP" profile, defined in [RFC3711]. However, it is expected that other secure RTP profiles (e.g., "RTP/SAVPF") can use the same descriptions, which are in accordance with the SRTP protocol specification [RFC3711].

There is no assurance that an endpoint is capable of configuring its SRTP service with a particular crypto attribute parameter, but SRTP guarantees minimal interoperability among SRTP endpoints through the default SRTP parameters [RFC3711]. More capable SRTP endpoints support a variety of parameter values beyond the SRTP defaults, and these values can be configured by the SRTP security descriptions defined here. An endpoint that does not support the crypto attribute will ignore it according to the SDP. Such an endpoint will not correctly process the particular media stream. By using the Offer/Answer model, the offerer and answerer can negotiate the crypto parameters to be used before commencement of the multimedia session (see Section 7.1).

There are over twenty cryptographic parameters listed in the SRTP specification. Many of these parameters have fixed values for particular cryptographic transforms. At the time of session establishment, however, there is usually no need to provide unique

settings for many of the SRTP parameters, such as salt length and pseudo-random function (PRF). Thus, it is possible to simplify the list of parameters by defining "cryptographic suites" that fix a set of SRTP parameter values for the security session. This approach is followed by the SRTP security descriptions, which uses the general security description parameters as follows:

* crypto-suite: Identifies the encryption and authentication transforms.
* key parameter: SRTP keying material and parameters
* session parameters: The following parameters are defined:
  - KDR: The SRTP Key Derivation Rate is the rate at which a pseudo-random function is applied to a master key.
  - UNENCRYPTED_SRTP: SRTP messages are not encrypted.
  - UNENCRYPTED_SRTCP: SRTCP messages are not encrypted.
  - UNAUTHENTICATED_SRTP: SRTP messages are not authenticated.
  - FEC_ORDER: Order of forward error correction (FEC) relative to SRTP services.
  - FEC_KEY: Master Key for FEC when the FEC stream is sent to a separate address and/or port.
  - WSH: Window Size Hint.
  - Extensions: Extension parameters can be defined.

Please refer to the SRTP specification for a complete list of parameters and their descriptions [Section 8.2, srtp]. Regarding the UNENCRYPTED_SRTCP parameter, offerers and answerers of SDP security descriptions MUST NOT use the SRTCP E-bit to override UNENCRYPTED_SRTCP or the default, which is to encrypt all SRTCP messages (see Section 6.3.2). The key parameter, the crypto-suite, and the session parameters shown above are described in detail in the following subsections.

6.1. SRTP Key Parameter

SRTP security descriptions define the use of the "inline" key method as described in the following. Use of any other keying method (e.g., URL) for SRTP security descriptions is for further study.

The "inline" type of key contains the keying material (master key and salt) and all policy related to that master key, including how long it can be used (lifetime) and whether it uses a master key identifier (MKI) to associate an incoming SRTP packet with a particular master key. Compliant implementations obey the policies associated with a master key and MUST NOT accept incoming packets that violate the policy (e.g., after the master key lifetime has expired).

The key parameter contains one or more cryptographic master keys, each of which MUST be a unique cryptographically random [RFC1750]

value with respect to other master keys in the entire SDP message (i.e., including master keys for other streams). Each key follows the format (the formal definition is provided in Section 9.2):

```
"inline:" <key||salt> ["|" lifetime] ["|" MKI ":" length]

key||salt      concatenated master key and salt, base64 encoded
               (see [RFC3548], Section 3)
lifetime       master key lifetime (max number of SRTP or SRTCP
               packets using this master key)
MKI:length     MKI and length of the MKI field in SRTP packets
```

The following definition provides an example for AES_CM_128_HMAC_SHA1_80:

```
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:4
```

The first field ("d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj") of the parameter is the cryptographic master key appended with the master salt; the two are first concatenated and then base64 encoded. The length of the concatenated key and salt is determined by the crypto-suite for which the key applies. If the length (after being decoded from base64) does not match that specified for the crypto-suite, the crypto attribute in question MUST be considered invalid. Each master key and salt MUST be a cryptographically random number and MUST be unique to the entire SDP message. When base64 decoding the key and salt, padding characters (i.e., one or two "=" at the end of the base64-encoded data) are discarded (see [RFC3548] for details). Base64 encoding assumes that the base64 encoding input is an integral number of octets. If a given crypto-suite requires the use of a concatenated key and salt with a length that is not an integral number of octets, said crypto-suite MUST define a padding scheme that results in the base64 input being an integral number of octets. For example, if the length defined were 250 bits, then 6 padding bits would be needed, which could be defined to be the last 6 bits in a 256 bit input.

The second field is the OPTIONAL lifetime of the master key as measured in maximum number of SRTP or SRTCP packets using that master key (i.e., the number of SRTP packets and the number of SRTCP packets each have to be less than the lifetime). The lifetime value MAY be written as a non-zero, positive decimal integer or as a power of 2 (see the grammar in Section 9.2 for details); leading zeroes MUST NOT be used. The "lifetime" value MUST NOT exceed the maximum packet lifetime for the crypto-suite. If the lifetime is too large or otherwise invalid, then the entire crypto attribute MUST be considered invalid. The default MAY be implicitly signaled by omitting the lifetime (note that the lifetime field never includes a

colon, whereas the third field always does). This is convenient when the SRTP cryptographic key lifetime is the default value. As a shortcut to avoid long decimal values, the syntax of the lifetime allows using the literal "2^", which indicates "two to the power of". The example above shows a case where the lifetime is specified as 2^20. The following example, which is for the AES_CM_128_HMAC_SHA1_80 crypto-suite, has a default for the lifetime field, which means that SRTP's and SRTCP's default values will be used (see [RFC3711]):

```
inline:YUJDZGVmZ2hpSktMbW9QUXJzVHVWd3l6MTIzNDU2|1066:4
```

The example shows a 30-octet key and concatenated salt that is base64 encoded: The 30-octet key/salt concatenation is expanded to 40 characters (octets) by the three-in-four encoding of base64.

The third field, which is also OPTIONAL, is the Master Key Identifier (MKI) and its byte length.

"MKI" is the master key identifier associated with the SRTP master key. The MKI is here defined as a positive decimal integer that is encoded as a big-endian integer in the actual SRTP packets; leading zeroes MUST NOT be used in the integer representation. If the MKI is given, then the length of the MKI MUST also be given and separated from the MKI by a colon (":"). The MKI length is the size of the MKI field in the SRTP packet, specified in bytes as a decimal integer; leading zeroes MUST NOT be used. If the MKI length is not given or its value exceeds 128 (bytes), then the entire crypto attribute MUST be considered invalid. The substring "1:4" in the first example assigns to the key a master key identifier of 1 that is 4 bytes long, and the second example assigns a 4-byte master key identifier of 1066 to the key. One or more master keys with their associated MKI can be initially defined, and then later updated, or deleted and new ones defined.

SRTP offers a second feature for specifying the lifetime of a master key in terms of two values, called "From" and "To," which are defined on the SRTP sequence number space [RFC3711]. This SRTP Security Descriptions specification, however, does not support the <"From", "To"> feature since the lifetime of an AES master key is 2^48 SRTP packets, which means that there is no cryptographic reason to replace a master key for practical point-to-point applications. For this reason, there is no need to support two means for signaling key update. The MKI is chosen over <"From", "To"> by this specification for the very few applications that need it since the MKI feature is simpler (though the MKI adds additional bytes to each packet, whereas <"From", "To"> does not).

As mentioned above, the key parameter can contain one or more master keys. When the key parameter contains more than one master key, all the master keys in that key parameter MUST include an MKI value.

When using the MKI, the MKI length MUST be the same for all keys in a given crypto attribute.

6.2. Crypto-Suites

The SRTP crypto-suites define the encryption and authentication transforms to be used for the SRTP media stream. The SRTP specification has defined three crypto-suites, which are described further in the following subsections in the context of the SRTP security descriptions. The table below provides an overview of the crypto-suites and their parameters:

	AES_CM_128_ HMAC_SHA1_80	AES_CM_128_ HMAC_SHA1_32	F8_128_ HMAC_SHA1_80
Master key length	128 bits	128 bits	128 bits
Master salt length	112 bits	112 bits	112 bits
SRTP lifetime	2^48 packets	2^48 packets	2^48 packets
SRTCP lifetime	2^31 packets	2^31 packets	2^31 packets
Cipher	AES Counter Mode	AES Counter Mode	AES F8 Mode
Encryption key	128 bits	128 bits	128 bits
MAC	HMAC-SHA1	HMAC-SHA1	HMAC-SHA1
SRTP auth. tag	80 bits	32 bits	80 bits
SRTCP auth. tag	80 bits	80 bits	80 bits
SRTP auth. key len.	160 bits	160 bits	160 bits
SRTCP auth. key len.	160 bits	160 bits	160 bits

6.2.1. AES_CM_128_HMAC_SHA1_80

AES_CM_128_HMAC_SHA1_80 is the SRTP default AES Counter Mode cipher and HMAC-SHA1 message authentication with an 80-bit authentication tag. The master-key length is 128 bits and has a default lifetime of a maximum of 2^48 SRTP packets or 2^31 SRTCP packets, whichever comes first [Page 39, srtp].

> SRTP allows 2^48 SRTP packets or 2^31 SRTCP packets, whichever comes first. However, it is RECOMMENDED that automated key management allow easy and efficient rekeying at intervals far smaller than 2^31 packets given today's media rates or even HDTV media rates.

The SRTP and SRTCP encryption key lengths are 128 bits. The SRTP and SRTCP authentication key lengths are 160 bits (see Security Considerations in Section 8). The master salt value is 112 bits in length and the session salt value is 112 bits in length. The pseudo-random function (PRF) is the default SRTP pseudo-random function that uses AES Counter Mode with a 128-bit key length.

The length of the base64-decoded key and salt value for this crypto-suite MUST be 30 characters (i.e., 240 bits); otherwise, the crypto attribute is considered invalid.

6.2.2. AES_CM_128_HMAC_SHA1_32

This crypto-suite is identical to AES_CM_128_HMAC_SHA1_80 except that the authentication tag is 32 bits.

The length of the base64-decoded key and salt value for this crypto-suite MUST be 30 octets i.e., 240 bits; otherwise, the crypto attribute is considered invalid.

6.2.3. F8_128_HMAC_SHA1_80

This crypto-suite is identical to AES_CM_128_HMAC_SHA1_80 except that the cipher is F8 [RFC3711].

The length of the base64-decoded key and salt value for this crypto-suite MUST be 30 octets, i.e., 240 bits; otherwise the crypto attribute is considered invalid.

6.2.4. Adding New Crypto-Suite Definitions

If new transforms are added to SRTP, new definitions for those transforms SHOULD be given for the SRTP security descriptions and published in a Standards-Track RFC. Sections 6.2.1 through 6.2.3 illustrate how to define crypto-suite values for particular cryptographic transforms. Any new crypto-suites MUST be registered with IANA following the procedures in Section 10.

6.3. Session Parameters

SRTP security descriptions define a set of "session" parameters, which OPTIONALLY may be used to override SRTP session defaults for the SRTP and SRTCP streams. These parameters configure an RTP session for SRTP services. The session parameters provide session-specific information to establish the SRTP cryptographic context.

6.3.1. KDR=n

KDR specifies the Key Derivation Rate, as described in Section 4.3.1 of [RFC3711].

The value n MUST be a decimal integer in the set {1,2,...,24}, which denotes a power of 2 from 2^1 to 2^24, inclusive; leading zeroes MUST NOT be used. The SRTP key derivation rate controls how frequently a new session key is derived from an SRTP master key(s) [RFC3711] given in the declaration. When the key derivation rate is not specified (i.e., the KDR parameter is omitted), a single initial key derivation is performed [RFC3711].

In the offer/answer model, KDR is a declarative parameter.

6.3.2. UNENCRYPTED_SRTCP and UNENCRYPTED_SRTP

SRTP and SRTCP packet payloads are encrypted by default. The UNENCRYPTED_SRTCP and UNENCRYPTED_SRTP session parameters modify the default behavior of the crypto-suites with which they are used:

* UNENCRYPTED_SRTCP signals that the SRTCP packet payloads are not encrypted.

* UNENCRYPTED_SRTP signals that the SRTP packet payloads are not encrypted.

In the offer/answer model, these parameters are negotiated. If UNENCRYPTED_SRTCP is signaled for the session, then the SRTCP E bit MUST be clear (0) in all SRTCP messages. If the default is used, all SRTCP messages are encrypted, and the E bit MUST be set (1) on all SRTCP messages.

6.3.3. UNAUTHENTICATED_SRTP

SRTP and SRTCP packet payloads are authenticated by default. The UNAUTHENTICATED_SRTP session parameter signals that SRTP messages are not authenticated. Use of UNAUTHENTICATED_SRTP is NOT RECOMMENDED (see Security Considerations).

> The SRTP specification requires use of message authentication for SRTCP, but not for SRTP [RFC3711].

In the offer/answer model, this parameter is negotiated.

6.3.4. FEC_ORDER=order

FEC_ORDER signals the use of forward error correction for the RTP packets [RFC2733]. The forward error correction values for "order" are FEC_SRTP or SRTP_FEC. FEC_SRTP signals that FEC is applied before SRTP processing by the sender of the SRTP media and after SRTP processing by the receiver of the SRTP media; FEC_SRTP is the default. SRTP_FEC is the reverse processing.

In the offer/answer model, FEC_ORDER is a declarative parameter.

6.3.5. FEC_KEY=key-params

FEC_KEY signals the use of separate master key(s) for a Forward Error Correction (FEC) stream. The master key(s) are specified with the exact same format as the SRTP Key Parameter defined in Section 6.1, and the semantic rules are the same - in particular, the master key(s) MUST be different from all other master key(s) in the SDP. An FEC_KEY MUST be specified when the FEC stream is sent to a different IP-address and/or port than the media stream to which it applies (i.e., the "m=" line), e.g., as described in RFC 2733, Section 11.1. When an FEC stream is sent to the same IP-address and port as the media stream to which it applies, an FEC_KEY MUST NOT be specified. If an FEC_KEY is specified in this latter case, the crypto attribute in question MUST be considered invalid.

In the offer/answer model, FEC_KEY is a declarative parameter.

6.3.6. Window Size Hint (WSH)

SRTP defines the SRTP-WINDOW-SIZE [RFC3711, Section 3.3.2] parameter to protect against replay attacks. The minimum value is 64 [RFC3711]; however, this value may be considered too low for some applications (e.g., video).

The Window Size Hint (WSH) session parameter provides a hint for how big this window should be to work satisfactorily (e.g., based on sender knowledge of the number of packets per second). However, there might be enough information given in SDP attributes like "a=maxprate" [maxprate] and the bandwidth modifiers to allow a receiver to derive the parameter satisfactorily. Consequently, this value is only considered a hint to the receiver of the SDP that MAY choose to ignore the value provided. The value is a decimal integer; leading zeroes MUST NOT be used.

In the offer/answer model, WSH is a declarative parameter.

6.3.7. Defining New SRTP Session Parameters

New SRTP session parameters for the SRTP security descriptions can be defined in a Standards-Track RFC and registered with IANA according to the registration procedures defined in Section 10.

New SRTP session parameters are by default mandatory. A newly defined SRTP session parameter that is prefixed with the dash character ("-"), however, is considered optional and MAY be ignored. If an SDP crypto attribute is received with an unknown session parameter that is not prefixed with a "-" character, that crypto attribute MUST be considered invalid.

6.4. SRTP Crypto Context Initialization

In addition to the various SRTP parameters defined above, there are three pieces of information that are critical to the operation of the default SRTP ciphers:

* SSRC: Synchronization source
* ROC: Roll-over counter for a given SSRC
* SEQ: Sequence number for a given SSRC

In a unicast session, as defined here, there are three constraints on these values.

The first constraint is on the SSRC, which makes an SRTP keystream unique from other participants. As explained in SRTP, the keystream MUST NOT be reused on two or more different pieces of plaintext. Keystream reuse makes the ciphertext vulnerable to cryptanalysis. One vulnerability is that known-plaintext fields in one stream can expose portions of the reused keystream, and this could further expose more plaintext in other streams. Since all current SRTP encryption transforms use keystreams, key sharing is a general problem [RFC3711]. SRTP mitigates this problem by including the SSRC of the sender in the keystream. But SRTP does not solve this problem in its entirety because the Real-time Transport Protocol has SSRC collisions, which although very rare [RFC3550] are quite possible. During a collision, two or more SSRCs that share a master key will have identical keystreams for overlapping portions of the RTP sequence number space. SRTP Security Descriptions avoid keystream reuse by making unique master keys REQUIRED for the sender and receiver of the security description. Thus, the first constraint is satisfied.

Also note that there is a second problem with SSRC collisions: the SSRC is used to identify the crypto context and thereby the cipher, key, ROC, etc. to process incoming packets. In case of

SSRC collisions, crypto context identification becomes ambiguous and correct packet processing may not occur. Furthermore, if an RTCP BYE packet is to be sent for a colliding SSRC, that packet may also have to be secured. In a (unicast) point-to-multipoint scenario, this can be problematic for the same reasons, i.e., it is not known which of the possible crypto contexts to use. Note that these problems are not unique to the SDP security descriptions; any use of SRTP needs to consider them.

The second constraint is that the ROC MUST be zero at the time that each SSRC commences sending packets. Thus, there is no concept of a "late joiner" in SRTP security descriptions, which are constrained to be unicast and pairwise. The ROC and SEQ form a "packet index" in the default SRTP transforms and the ROC is consistently set to zero at session commencement, according to this document.

The third constraint is that the initial value of SEQ SHOULD be chosen to be within the range of 0..2^15-1; this avoids an ambiguity when packets are lost at the start of the session. If it is at the start of a session, an SSRC source might randomly select a high sequence-number value and put the receiver in an ambiguous situation: if initial packets are lost in transit up to the point that the sequence number wraps (i.e., exceeds 2^16-1), then the receiver might not recognize that its ROC needs to be incremented. By restricting the initial SEQ to the range of 0..2^15-1, SRTP packet-index determination will find the correct ROC value, unless all the first 2^15 packets are lost (which seems, if not impossible, rather unlikely). See Section 3.3.1 of the SRTP specification regarding packet-index determination [RFC3711].

6.4.1. Late Binding of One or More SSRCs to a Crypto Context

The packet index, therefore, depends on the SSRC, the SEQ of an incoming packet, and the ROC, which is an SRTP crypto context variable. Thus, SRTP has a big security dependency on SSRC uniqueness.

Given the above constraints, unicast SRTP crypto contexts can be established without the need to negotiate SSRC values in the SRTP security descriptions. Instead, an approach called "late binding" is RECOMMENDED by this specification. When a packet arrives, the SSRC that is contained in it can be bound to the crypto context at the time of session commencement (i.e., SRTP packet arrival) rather than at the time of session signaling (i.e., receipt of an SDP). With the arrival of the packet containing the SSRC, all the data items needed for the SRTP crypto context are held by the receiver. (Note that the ROC value by definition is zero; if non-zero values were to be supported, additional signaling would be required.) In other words,

the crypto context for a secure RTP session using late binding is initially identified by the SDP as

<*, address, port>

where '*' is a wildcard SSRC, "address" is the local receive address from the "c=" line, and "port" is the local receive port from the "m=" line. When the first packet arrives with ssrcX in its SSRC field, the crypto context

<ssrcX, address, port>

is instantiated subject to the following constraints:

* Media packets are authenticated: authentication MUST succeed; otherwise, the crypto context is not instantiated.

* Media packets are not authenticated: crypto context is automatically instantiated.

Note that use of late binding when there is no authentication of the SRTP media packets is subject to numerous security attacks, and that consequently it is NOT RECOMMENDED (of course, this can be said for unauthenticated SRTP in general).

Note that use of late binding without authentication will result in the creation of local state as a result of receiving a packet from any unknown SSRC. UNAUTHENTICATED_SRTP, therefore, is NOT RECOMMENDED because it invites easy denial-of-service attack. In contrast, late binding with authentication does not suffer from this weakness.

6.4.2. Sharing Cryptographic Contexts among Sessions or SSRCs

With the constraints and procedures described above, it is not necessary to explicitly signal the SSRC, ROC, and SEQ for a unicast RTP session. So there are no a=crypto parameters for signaling SSRC, ROC, or SEQ. Thus, multiple SSRCs from the same entity will share a=crypto parameters when late binding is used. Multiple SSRCs from the same entity arise due to either multiple sources (microphones, cameras, etc.) or RTP payloads requiring SSRC multiplexing within that same session. SDP also allows multiple RTP sessions to be defined in the same media description ("m="); these RTP sessions will also share the a=crypto parameters. An application that uses a=crypto in this way serially shares a master key among RTP sessions or SSRCs and MUST replace the master key when the aggregate number of packets among all SSRCs approaches 2^31 packets. SSRCs that share a master key MUST be unique from one another.

6.5. Removal of Crypto Contexts

The mechanism defined above addresses the issue of creating crypto contexts. However, in practice, session participants may want to remove crypto contexts prior to session termination. Since a crypto context contains information that cannot automatically be recovered (e.g., ROC), it is important that the sender and receiver agree on when a crypto context can be removed, and perhaps more importantly when it cannot.

> Even when late binding is used for a unicast stream, the ROC is lost and cannot be recovered automatically (unless it is zero) once the crypto context is removed.

We resolve this problem as follows. When SRTP security descriptions are being used, crypto-context removal MUST follow the same rules as SSRC removal from the member table [RFC3550]; note that this can happen as the result of an SRTCP BYE packet or a simple time-out due to inactivity. Inactive session participants that wish to ensure their crypto contexts are not timed out MUST thus send SRTCP packets at regular intervals.

7. SRTP-Specific Use of the Crypto Attribute

Section 5 describes general use of the crypto attribute, and this section completes it by describing SRTP-specific use.

7.1. Use with Offer/Answer

In this section, we describe how the SRTP security descriptions are used with the offer/answer model to negotiate cryptographic capabilities and communicate SRTP master keys. The rules defined below complement the general offer/answer rules defined in Section 5.1, which MUST be followed, unless otherwise specified. Note that the rules below define unicast operation only; support for multicast and multipoint unicast streams is for further study.

7.1.1. Generating the Initial Offer - Unicast Streams

When the initial offer is generated, the offerer MUST follow the steps in Section 5.1.1, as well as the following steps.

For each unicast media line (m=) using the secure RTP transport where the offerer wants to specify cryptographic parameters, the offerer MUST provide at least one valid SRTP security description ("a=crypto" line), as defined in Section 6. If the media stream includes Forward

Error Correction with a different IP-address and/or port from that of the media stream itself, an FEC_KEY parameter MUST be included, as described in Section 6.3.5.

The inline parameter conveys the SRTP master key used by an endpoint to encrypt the SRTP and SRTCP streams transmitted by that endpoint. The same key is used by the recipient to decrypt those streams. However, the receiver MUST NOT use that same key for the SRTP or SRTCP packets that it sends to the session because the default SRTP cipher and mode is insecure when the master key is reused across distinct SRTP streams.

The offerer MAY include one or more other SRTP session parameters, as defined in Section 6.3. Note, however, that if any SRTP session parameters are included that are not known to the answerer, but that are nonetheless mandatory (see Section 6.3.6), the negotiation will fail if the answerer does not support them.

7.1.2. Generating the Initial Answer - Unicast Streams

When the initial answer is generated, the answerer MUST follow the steps in Section 5.1.2, as well as the following steps.

For each unicast media line that uses the secure RTP transport and contains one or more "a=crypto" lines in the offer, the answerer MUST either accept one (and only one) of the crypto lines for that media stream, or it MUST reject the media stream. Only "a=crypto" lines that are considered valid SRTP security descriptions, as defined in Section 6, can be accepted. Furthermore, all parameters (crypto-suite, key parameter, and mandatory session parameters) MUST be acceptable to the answerer in order for the offered media stream to be accepted. Note that if the media stream includes Forward Error Correction with a different IP-address and/or port from that of the media stream itself, an FEC_KEY parameter MUST be included, as described in Section 6.3.5.

When the answerer accepts an SRTP unicast media stream with a crypto line, the answerer MUST include one or more master keys appropriate for the selected crypto algorithm; the master key(s) included in the answer MUST be different from those in the offer.

> When the master key(s) are not shared between the offerer and answerer, SSRC collisions between the offerer and answerer will not lead to keystream reuse, and hence SSRC collisions do not necessarily have to be prevented.

If Forward Error Correction to a separate IP-address and/or port is included, the answer MUST include an FEC_KEY parameter, as described in Section 6.3.5.

Declarative session parameters may be added to the answer as usual; however, the answerer SHOULD NOT add any mandatory session parameter (see Section 6.3.6) that might be unknown to the offerer.

If the answerer cannot find any valid crypto line that it supports, or if its configured policy prohibits any cryptographic key parameter (e.g., key length) or cryptographic session parameter (e.g., KDR, FEC_ORDER), it MUST reject the media stream, unless it is able to successfully negotiate use of SRTP by other means outside the scope of this document (e.g., by use of MIKEY [mikey]).

7.1.3. Processing of the Initial Answer - Unicast Streams

When the offerer receives the answer, it MUST perform the steps in Section 5.1.3, as well as the following steps for each SRTP media stream it offered with one or more crypto lines in it.

If the media stream was accepted and it contains a crypto line, it MUST be checked that the crypto line is valid according to the constraints specified in Section 6 (including any FEC constraints).

If the offerer either does not support or is not willing to honor one or more of the SRTP parameters in the answer, the offerer MUST consider the crypto line invalid.

If the crypto line is not valid, or the offerer’s configured policy prohibits any cryptographic key parameter (e.g., key length) or cryptographic session parameter, the SRTP security negotiation MUST be deemed to have failed.

7.1.4. Modifying the Session

When a media stream using the SRTP security descriptions has been established and a new offer/answer exchange is performed, the offerer and answerer MUST follow the steps in Section 5.1.4, as well as the following steps.

When modifying the session, all negotiated aspects of the SRTP media stream can be modified. For example, a new crypto suite can be used or a new master key can be established. As described in RFC 3264, when a new offer/answer exchange is made, there will be a window of time where the offerer and the answerer must be prepared to receive media according to both the old and new offer/answer exchange.

This requirement applies here as well; however, the following should be noted:

* When authentication is not being used, it may not be possible for either the offerer or answerer to determine if a given packet is encrypted according to the old or new offer/answer exchange. RFC 3264 defines a couple of techniques to address this problem, e.g., changing the payload types used and/or the transport addresses. Note, however, that a change in transport addresses may have an impact on quality of service as well as on firewall and NAT traversal. The SRTP security descriptions use the MKI to deal with this (which adds a few bytes to each SRTP packet), as described in Section 6.1. For further details on the MKI, please refer to [RFC3711].

* If the answerer changes its master key, the offerer will not be able to process packets secured via this master key until the answer is received. This could be addressed by using a security "precondition" [sprecon].

If the offerer includes an IP address and/or port that differs from that used previously for a media stream (or FEC stream), the offerer MUST include a new master key with the offer (and in so doing, it will be creating a new crypto context where the ROC is set to zero). Similarly, if the answerer includes an IP address and/or port that differs from that used previously for a media stream (or FEC stream), the answerer MUST include a new master key with the answer (and hence create a new crypto context with the ROC set to zero). The reason for this is that when the answerer receives an offer or the offerer receives an answer with an updated IP address and/or port, it is not possible to determine if the other side has access to the old crypto context parameters (and in particular the ROC). For example, if one side is a decomposed media gateway, or if a SIP back-to-back user agent is involved, it is possible that the media endpoint changed and no longer has access to the old crypto context. By always requiring a new master key in this case, the answerer/offerer will know that the ROC is zero for this offer/answer, and any key lifetime constraints will trivially be satisfied too. Another consideration here applies to media relays; if the relay changes the media endpoint on one side transparently to the other side, the relay cannot operate as a simple packet reflector but will have to actively engage in SRTP packet processing and transformation (i.e., decryption and re-encryption, etc.).

Finally, note that if the new offer is rejected, the old crypto parameters remain in place.

7.1.5. Offer/Answer Example

In this example, the offerer supports two crypto suites (f8 and AES). The a=crypto line is actually one long line, although it is shown as two lines in this document due to page formatting. The f8 example shows two inline parameters; as explained in Section 6.1, there may be one or more key (i.e., inline) parameters in a crypto attribute. In this way, multiple keys are offered to support key rotation using a Master Key Identifier (MKI).

Offerer sends:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
 inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
 FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
 inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
 inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
 FEC_ORDER=FEC_SRTP
```

Answerer replies:

```
v=0
o=jill 25690844 8070842634 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=homer@example.com (Homer Simpson)
c=IN IP4 168.2.17.11
t=2873397526 2873405696
m=audio 32640 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
 inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

In this case, the session would use the AES_CM_128_HMAC_SHA1_80 crypto suite for the RTP and RTCP traffic. If F8_128_HMAC_SHA1_80 were selected by the answerer, there would be two inline keys associated with the SRTP cryptographic context. One key has an MKI value of 1 and the second has an MKI of 2.

7.2. SRTP-Specific Use Outside Offer/Answer

Use of SRTP security descriptions outside the offer/answer model is not defined.

Use of SRTP security descriptions outside the offer/answer model could have been defined for sendonly media streams; however, there would not be a way to indicate the key to use for SRTCP by the receiver of said media stream.

7.3. Support for SIP Forking

As mentioned earlier, the security descriptions defined here do not support multicast media streams or multipoint unicast streams. However, in the SIP protocol, it is possible to receive several answers to a single offer due to the use of forking (see [SIP]). Receiving multiple answers leads to a couple of problems for the SRTP security descriptions:

* Different answerers may choose different ciphers, keys, etc.; however, there is no way for the offerer to associate a particular incoming media packet with a particular answer.

* Two or more answerers may pick the same SSRC, and hence the SSRC collision problems mentioned earlier may arise.

As stated earlier, the above point-to-multipoint cases are outside the scope of the SDP security descriptions. However, there are still ways of supporting SIP forking, e.g., by changing the multipoint scenario resulting from SIP forking into multiple two-party unicast cases. This can be done as follows:

For each answer received beyond the initial answer, issue a new offer to that particular answerer using a new receive transport address (IP address and port); note that this requires support for the SIP UPDATE method [RFC3311]. Also, to ensure that two media sessions are not inadvertently established prior to the UPDATE being processed by one of them, use security preconditions [sprecon].

Finally, note that all SIP User Agents that received the offer will know the key(s) being proposed by the initial offer. If the offerer wants to ensure security with respect to all other User Agents that may have received the offer, a new offer/answer exchange with a new key needs to be performed with the answerer as well. Note that the offerer cannot determine whether a single or multiple SIP User Agents received the offer, since intermediate forking proxies may only forward a single answer to the offerer.

The above description is intended to suggest one possible way of supporting SIP forking. There are many details missing and it should not be considered a normative specification. Alternative approaches may also be possible

7.4. SRTP-Specific Backwards Compatibility Considerations

It is possible that the answerer supports the SRTP transport and accepts the offered media stream, but that it does not support the crypto attribute defined here. The offerer can recognize this situation by seeing an accepted SRTP media stream in the answer that does not include a crypto line. In that case, the security negotiation defined here MUST be deemed to have failed.

Also, if a media stream with a given SRTP transport (e.g., "RTP/SAVP") is sent to a device that does not support SRTP, that media stream will be rejected.

7.5. Operation with KEYMGT= and k= lines

An offer MAY include both "a=crypto" and "a=keymgt" lines [keymgt]. Per SDP rules, the answerer will ignore attribute lines that it does not understand. If the answerer supports both "a=crypto" and "a=keymgt", the answer MUST include either "a=crypto" or "a=keymgt", but not both, as including both is undefined.

An offer MAY include both "a=crypto" and "k=" lines [RFC4566]. Per SDP rules, the answerer will ignore attribute lines it does not understand. If the answerer supports both "a=crypto" and "k=", the answer MUST include either "a=crypto" or "k=" but not both, as including both is undefined.

8. Security Considerations

Like all SDP messages, SDP messages containing security descriptions are conveyed in an encapsulating application protocol (e.g., SIP, MGCP). It is the responsibility of the encapsulating protocol to ensure the protection of the SDP security descriptions. Therefore, IT IS REQUIRED that the application invoke its own security mechanisms (e.g., secure multiparts such as S/MIME [smime]) or, alternatively, utilize a lower-layer security service (e.g., TLS or IPsec). IT IS REQUIRED that this security service provide strong message authentication and packet-payload encryption, as well as effective replay protection.

"Replay protection" is needed against an attacker that has enough access to the communications channel to intercept messages and to deliver copies to the destination. A successful replay attack will

cause the recipient to perform duplicate processing on a message; the attack is worse when the duped recipient sends a duplicate reply to the initiator. Replay protections are not found in S/MIME or in the other secure-multiparts standard, PGP/MIME. S/MIME and PGP/MIME, therefore, need to be augmented with some replay-protection mechanism that is appropriate to the encapsulating application protocol (e.g., SIP, MGCP). Three common ways to provide replay protection are to place a sequence number in the message, to use a timestamp, or for the receiver to keep a hash of the message to be compared with incoming messages. There typically needs to be a replay "window" and some policy for keeping state information from previous messages in a "replay table" or list.

The discussion that follows uses "message authentication" and "message confidentiality" in a manner consistent with SRTP [RFC3711]. "Message confidentiality" means that only the holder of the secret decryption key can access the plain-text content of the message. The decryption key is the same key as the encryption key, using SRTP counter mode and f8 encryption transforms, which are vulnerable to message tampering and need SRTP message authentication to detect such tampering. "Message authentication" and "message integrity validation" generally mean the same thing in IETF security standards: an SRTP message is authenticated following a successful HMAC integrity check [RFC3711], which proves that the message originated from the holder of an SRTP master key and was not altered en route. Such an "authentic" message, however, can be captured by an attacker and "replayed" when the attacker re-inserts the packet into the channel. A replayed packet can have a variety of bad effects on the session, and SRTP uses the extended sequence number to detect replayed SRTP packets [RFC3711].

The SRTP specification identifies which services and features are default values that are normative-to-implement (such as AES_CM_128_80) versus normative-to-use (such as AES_CM_128_32).

8.1. Authentication of Packets

Security descriptions as defined herein signal security services for RTP packets. RTP messages are vulnerable to a variety of attacks, such as replay and forging. To limit these attacks, SRTP message integrity mechanisms SHOULD be used (SRTP replay protection is always enabled).

8.2. Keystream Reuse

SRTP security descriptions signal configuration parameters for SRTP sessions. Misconfigured SRTP sessions are vulnerable to attacks on their encryption services when running the crypto suites defined in

Sections 6.2.1, 6.2.2, and 6.2.3. An SRTP encryption service is "misconfigured" when two or more media streams are encrypted using the same keystream of AES blocks. When senders and receivers share derived session keys, SRTP requires that the SSRCs of session participants serve to make their corresponding keystreams unique, which is violated in the case of SSRC collision: SRTP SSRC collision drastically weakens SRTP or SRTCP payload encryption during the time that identical keystreams are used [RFC3711]. An attacker, for example, might collect SRTP and SRTCP messages and await a collision. This attack on the AES-CM and AES-f8 encryption is avoided entirely when each media stream has its own unique master key in both the send and receive direction. This specification restricts use of SDP security description to unicast point-to-point streams so that keys are not shared between SRTP hosts, and the master keys used in the send and receive direction for a given media stream are unique.

8.3. Signaling Authentication and Signaling Encryption

There is no reason to incur the complexity and computational expense of SRTP, however, when its key establishment is exposed to unauthorized parties. In most cases, the SRTP crypto attribute and its parameters are vulnerable to denial-of-service attacks when they are carried in an unauthenticated SDP message. In some cases, the integrity or confidentiality of the RTP stream can be compromised. For example, if an attacker sets UNENCRYPTED for the SRTP stream in an offer, this could result in the answerer’s not decrypting the encrypted SRTP messages. In the worst case, the answerer might itself send unencrypted SRTP and leave its data exposed to snooping.

Thus, IT IS REQUIRED that MIME secure multiparts, IPsec, TLS, or some other data security service be used to provide message authentication for the encapsulating protocol that carries the SDP messages having a crypto attribute (a=crypto). Furthermore, IT IS REQUIRED that encryption of the encapsulating payload be used whenever a master key parameter (inline) appears in the message. Failure to encrypt the SDP message containing an inline SRTP master key renders the SRTP authentication or encryption service useless in practically all circumstances. Failure to authenticate an SDP message that carries SRTP parameters renders the SRTP authentication or encryption service useless in most practical applications.

When the communication path of the SDP message is routed through intermediate systems that inspect parts of the SDP message, security protocols such as [IPsec] or TLS SHOULD NOT be used for encrypting and/or authenticating the security description. In the case of intermediate-system processing of a message containing SDP security descriptions, the "a=crypto" attributes SHOULD be protected end-to-end so that the intermediate system can neither modify the security

description nor access the keying material. Network or transport security protocols that terminate at each intermediate system, therefore, SHOULD NOT be used for protecting SDP security descriptions. A security protocol SHOULD allow the security descriptions to be encrypted and authenticated end-to-end independently of the portions of the SDP message that any intermediate system modifies or inspects: MIME secure multiparts are RECOMMENDED for the protection of SDP messages that are processed by intermediate systems.

9. Grammar

In this section, we first provide the ABNF grammar for the generic crypto attribute, and then we provide the ABNF grammar for the SRTP-specific use of the crypto attribute.

9.1. Generic "Crypto" Attribute Grammar

The ABNF grammar for the crypto attribute is defined below:

```
"a=crypto:" tag 1*WSP crypto-suite 1*WSP key-params
                                        *(1*WSP session-param)

tag              = 1*9DIGIT
crypto-suite     = 1*(ALPHA / DIGIT / "_")

key-params       = key-param *(";" key-param)
key-param        = key-method ":" key-info
key-method       = "inline" / key-method-ext
key-method-ext   = 1*(ALPHA / DIGIT / "_")
key-info         = 1*(%x21-3A / %x3C-7E) ; visible (printing) chars
                                         ; except semi-colon
session-param    = 1*(VCHAR)             ; visible (printing) characters
```

where WSP, ALPHA, DIGIT, and VCHAR are defined in [RFC4234].

9.2. SRTP "Crypto" Attribute Grammar

This section provides an Augmented BNF [RFC4234] grammar for the SRTP-specific use of the SDP crypto attribute:

```
   crypto-suite        = srtp-crypto-suite
   key-method          = srtp-key-method
   key-info            = srtp-key-info
   session-param       = srtp-session-param

   srtp-crypto-suite   = "AES_CM_128_HMAC_SHA1_32" /
                         "F8_128_HMAC_SHA1_32" /
```

```
                          "AES_CM_128_HMAC_SHA1_80" /
                          srtp-crypto-suite-ext

      srtp-key-method     = "inline"
      srtp-key-info       = key-salt ["|" lifetime] ["|" mki]

      key-salt            = 1*(base64)   ; binary key and salt values
                                         ; concatenated together, and then
                                         ; base64 encoded [section 3 of
                                         ; RFC3548

      lifetime           = ["2^"] 1*(DIGIT)   ; see section 6.1 for "2^"
      mki                 = mki-value ":" mki-length
      mki-value           = 1*DIGIT
      mki-length          = 1*3DIGIT   ; range 1..128.

      srtp-session-param  = kdr /
                            "UNENCRYPTED_SRTP" /
                            "UNENCRYPTED_SRTCP" /
                            "UNAUTHENTICATED_SRTP" /
                            fec-order /
                            fec-key /
                            wsh /
                            srtp-session-extension

      kdr                 = "KDR=" 1*2(DIGIT)  ; range 0..24,
                                               ; power of two

      fec-order           = "FEC_ORDER=" fec-type
      fec-type            = "FEC_SRTP" / "SRTP_FEC"
      fec-key             = "FEC_KEY=" key-params

      wsh                 = "WSH=" 2*DIGIT    ; minimum value is 64
      base64              =  ALPHA / DIGIT / "+" / "/" / "="

      srtp-crypto-suite-ext  = 1*(ALPHA / DIGIT / "_")
      srtp-session-extension = ["-"] 1*(VCHAR)  ;visible chars [RFC4234]
                               ; first character must not be dash ("-")
```

10. IANA Considerations

10.1. Registration of the "crypto" Attribute

The IANA has registered a new SDP attribute as follows:

Attribute name:	crypto
Long form name:	Security description cryptographic attribute for media streams
Type of attribute:	Media-level
Subject to charset:	No
Purpose:	Security descriptions
Appropriate values:	See Section 4

10.2. New IANA Registries and Registration Procedures

The following sub-sections define a new IANA registry with associated sub-registries to be used for the SDP security descriptions. The IANA has created an SDP Security Description registry as shown below and further described in the following sections:

```
SDP Security Descriptions
  |
  +- Key Methods (described in 10.2.1)
  |
  +- Media Stream Transports (described in 10.2.2)
       |
       +- Transport1 (e.g., SRTP)
       |    |
       |    +- Supported Key Methods (e.g., inline)
       |    |
       |    +- crypto suites
       |    |
       |    +- session parameters
       |
       +- Transport2
       :    :
```

10.2.1. Key Method Registry and Registration

The IANA has created a new subregistry for SDP security description key methods. An IANA key method registration MUST be documented in an RFC in accordance with the [RFC2434] Standards Action, and it MUST provide the name of the key method in accordance with the grammar for key-method-ext defined in Section 9.1.

10.2.2. Media Stream Transport Registry and Registration

The IANA has created a new subregistry for SDP security description Media Stream Transports. An IANA media stream transport registration MUST be documented in an RFC in accordance with the RFC 2434 Standards Action and the procedures defined in Sections 4 and 5 of this document. The registration MUST provide the name of the transport and a list of supported key methods.

In addition, each new media stream transport registry must contain a crypto-suite registry and a session parameter registry, as well as IANA instructions for how to populate these registries.

10.3. Initial Registrations

10.3.1. Key Method

The following security descriptions key methods are hereby registered:

inline

10.3.2. SRTP Media Stream Transport

The IANA has created an SDP Security Description Media Stream Transport subregistry for "SRTP". The key methods supported is "inline". The reference for the SDP security description for SRTP is this document.

10.3.2.1. SRTP Crypto Suite Registry and Registration

The IANA has created a new subregistry for SRTP crypto suites under the SRTP transport of the SDP Security Descriptions. An IANA SRTP crypto suite registration MUST indicate the crypto suite name in accordance with the grammar for srtp-crypto-suite-ext defined in Section 9.2.

The semantics of the SRTP crypto suite MUST be described in an RFC in accordance with the RFC 2434 Standards Action, including the semantics of the "inline" key-method and any special semantics of parameters.

The following SRTP crypto suites are hereby registered:

AES_CM_128_HMAC_SHA1_80
AES_CM_128_HMAC_SHA1_32
F8_128_HMAC_SHA1_80

The reference for these crypto suites is provided in this document.

10.3.2.2. SRTP Session Parameter Registration

The IANA has created a new subregistry for SRTP session parameters under the SRTP transport of the SDP Security Descriptions. An IANA SRTP session parameter registration MUST indicate the session parameter name (srtp-session-extension as defined in Section 9.2); the name MUST NOT begin with the dash character ("-").

The semantics of the parameter MUST be described in an RFC in accordance with the RFC 2434 Standards Action. If values can be assigned to the parameter, then the format and possible values that can be assigned MUST be described in the RFC in accordance with the RFC 2434 Standards Action as well. Also, it MUST be specified whether the parameter is declarative or negotiated in the offer/answer model.

The following SRTP session parameters are hereby registered:

```
KDR
UNENCRYPTED_SRTP
UNENCRYPTED_SRTCP
UNAUTHENTICATED_SRTP
FEC_ORDER
FEC_KEY
WSH
```

The reference for these parameters is this document.

11. Acknowledgements

This document is a product of the IETF MMUSIC working group and has benefited from comments from its participants. This document also benefited from discussions with Elisabetta Cararra, Earl Carter, Per Cederqvist, Bill Foster, Matt Hammer, Cullen Jennings, Paul Kyzivat, David McGrew, Mats Naslund, Dave Oran, Jonathan Rosenberg, Dave Singer, Mike Thomas, Brian Weis, and Magnus Westerlund.

12. Normative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC2828] Shirey, R., "Internet Security Glossary", FYI 36, RFC 2828, May 2000.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC1750] Eastlake 3rd, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC3548] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

13. Informative References

[sprecon] Andreasen, F. and D. Wing, "Security Preconditions for Session Description Protocol Media Streams", Work in Progress, October 2005.

[RFC3407] Andreasen, F., "Session Description Protocol (SDP) Simple Capability Declaration", RFC 3407, October 2002.

[Bellovin] 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.

[GDOI] Baugher, M., Weis, B., Hardjono, T., and H. Harney, "The Group Domain of Interpretation", RFC 3547, July 2003.

[kink] Sakane, S., Kamada, K., Thomas, M. and J. Vilhuber, "Kerberized Internet Negotiation of Keys (KINK)", RFC 4430, March 2006.

[ike] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[ipsec] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[maxprate] Westerlund, M., "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP)", RFC 3890, September 2004.

[RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

[s/mime] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[pgp/mime] Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, October 1996.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[keymgt] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[mikey] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[skeme] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for the Internet", ISOC Secure Networks and Distributed Systems Symposium, San Diego, 1996.

[RFC3312] Camarillo, G., Marshall, W., and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312, October 2002.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[srtpf] Ott, J. and E. Carrara, "Extended Secure RTP Profile for RTCP-based Feedback (RTP/SAVPF)", work in progress, October 2003.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, September 2002.

Appendix A - Rationale for Keying Material Directionality

SDP security descriptions define the keying material for the sending direction, which is included in the SDP. Thus, the key that is carried in an SDP message is a decryption key for the receiver of that SDP message. This is in contrast to the majority of information included in SDP, which describes information for the receiving (or receiving and sending) direction. This reversed information directionality generates some challenges with using the mechanism in the offer/answer model and in particular with SIP, where early media and forking require special consideration (as described in Section 7.3). There are however good reasons for why this was done, which can be summarized as follows:

First of all, there is the general security philosophy of letting the entity that sends traffic decide what key to use for protecting it. SRTP uses counter mode, which is secure when counters do not overlap among senders who share a master key; the surest way to avoid counter overlap is for each endpoint to generate its own master key. Secondly, if SDP security descriptions had been designed to keep the normal SDP information directionality, it would have resulted in problems with supporting early media and SIP forking: If an offer generates multiple answers and the keying material was for the receive direction, some of the parameter values (e.g. lifetime) would have to be shared between all the answerers (senders of media), which would lead to considerable complexity, possibly requiring changes or extensions to SRTP. Other problems were discovered as well, which we describe further below.

In the following scenarios, we analyze what would occur if SDP security descriptions had been designed so that the keying material was the receive keying material (rather than its actual design, where the keying material is the sending keying material):

Scenario A: Non-Forking Case

In this scenario, the offer includes the receiving keying material, the answerer receives it and starts sending data packets towards the offerer. If there was a single crypto attribute in the offer, there would be no ambiguity about which crypto suite was being used and, hence, the incoming packet could be processed. However, in the case where the offer included multiple alternative crypto-attributes, the offerer would not know which one was chosen, and hence, if the offerer received packets before the answer came back, the offerer would be unable to process those packets (problem 1). (Use of the MKI has been suggested as one possible solution to that, however it incurs a per-packet overhead.)

Scenario B: Serial Forking Case

In this scenario, Alice generates an offer to Bob, who starts sending (early) media towards Alice (no answer returned yet). In this scenario, we assume we aren't also encountering Scenario A (e.g., the offer includes only a single crypto-attribute) and that Bob is using a Synchronization Source (SSRC) value of 1 for his SRTP and SRTCP packets. Alice thus has a crypto-context for SSRC 1, including the associated ROC (Roll Over Counter) and SEQ (RTP Sequence Number). Bob now forwards the call to Carol (Bob still has not generated an answer). At this point, Bob has Alice's key, which sometimes might be a security weakness. As the exchange proceeds, Carol gets the original offer, including the offered crypto-attribute and starts sending media packets towards Alice. It just so happens that Carol chooses an SSRC value of 1, as did Bob. When Carol starts generating packets, there is a potential for what RFC 3711 calls a "two-time pad" issue (problem 2), as well as the potential for the ROC to be out of sync between Alice and Carol (problem 3). Note that since Bob and Carol are (presumably) using different source transport addresses, the SSRC reuse does not constitute an SSRC collision (although it may still be interpreted as such by Alice). Per RFC 3711, since the master key would be shared between Bob and Carol in this case, it is RECOMMENDED that Alice leave the session at that point in order to avoid the two-time pad issue. It should also be noted that RFC 3711 recommends against sharing SRTP master keys, which forking may accidentally introduce when the keying material is for the receiving direction.

If we consider the above scenario again, but this time with keying material in the offer (and answer) being the sending keying material (as specified by SDP security descriptions), the scenario instead looks as follows: Bob again chooses SSRC 1, and Bob will

need to send back an answer to Alice, since Alice needs to learn Bob's sending key. Bob also starts sending media towards Alice (clipping may occur until Alice receives Bob's answer). Bob again forwards the call to Carol who also starts sending early media using SSRC 1. However, Carol needs to generate a new answer (for the dialog between Alice and Carol) in order for Alice to process Carol's packets . Upon receiving this answer, Alice can initiate a new offer/answer exchange (to move the session to another transport address as described in Section 7.3). In this case, there is one master key per session and a unique keystream regardless of whether or not SSRCs collide.

Scenario C: Parallel Forking Case

In this scenario, Alice generates an offer (with receive keying material) that gets forked to Bob and Carol in parallel. Bob and Carol both start sending packets (early media) to Alice. If Bob and Carol choose different SSRCs, everything is fine initially. However, one of the crypto context parameters is the master key lifetime, and since Bob and Carol are sharing the same master key (unbeknownst to either), they do not know when they need to rekey (problem 4). If they choose the same SSRC, we have the two-time pad problem again (problem 2).

In summary, if keying material were for the receive direction, we would have the following problems:

- Problem 1: Offerer does not know which of multiple crypto offers was chosen by answerer.

- Problem 2: SSRC reuse (or SSRC collisions) between multiple answerers (serial or parallel forking) may lead to the two-time pad issue.

- Problem 3: Part of the crypto context parameters (specifically the ROC) is not communicated but derived, and if we allow multiple entities to use the same SSRC (sequentially), the ROC can be wrong.

- Problem 4: All crypto contexts that share a master key need to maintain a shared set of counters (master key lifetime), and if we allow for multiple entities on different platforms to share a master key, we would need a mechanism to synchronize these counters.

Problem 1 could be addressed by using the MKI as proposed separately; however, it would result in using extra bandwidth for each SRTP media packet. Solving problem 2 implies a need for

being able to synchronize SSRC values with the answerer (or abandon the session when SSRC reuse or SSRC collisions occur). Problem 3 implies a need for being able to synchronize ROC values on a per SSRC basis (or abandon the session when SSRC reuse occurs). Problem 4 could be solved by having the offerer (Alice, i.e., the entity receiving media) determine how many packets have actually been generated by the total set of senders to Alice and, hence, be the one to initiate the rekeying. In the case of packet losses, etc. this is not foolproof, but in practice it could probably be addressed by use of a reasonable safety margin.

In conclusion, it would be expected from an offer/answer and SIP point of view to have the offer (and answer) keying material be the receive keying material; however, doing so would trade security for SIP friendliness, e.g., two-time pad and master key lifetime issues, and violate the RFC 3711 rule for sharing an SRTP master key across SRTP sessions.

Authors' Addresses

Flemming Andreasen
Cisco Systems, Inc.
499 Thornall Street, 8th Floor
Edison, New Jersey 08837 USA

EMail: fandreas@cisco.com

Mark Baugher
5510 SW Orchid Street
Portland, Oregon 97219 USA

EMail: mbaugher@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134 USA

EMail: dwing@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).