

Internet Engineering Task Force  
MMUSIC Working Group  
INTERNET-DRAFT  
Expires: August 2003

J. Arkko  
E. Carrara  
F. Lindholm  
M. Naslund  
K. Norrman  
Ericsson  
February, 2003

Key Management Extensions for SDP and RTSP  
<[draft-ietf-mmusic-kmgmt-ext-06.txt](#)>

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Abstract

This document defines general extensions for SDP and RTSP to carry the security information needed by a key management protocol, in order to secure the media. These extensions are presented as a framework, to be used by one or more key management protocols. As such, its use is meaningful only when it is completed by the key management protocol in use.

General guidelines are also given on how the framework should be used together with SIP and RTSP.

INTERNET-DRAFT

mmusic-kmgmt-ext-06

February 2003

## TABLE OF CONTENTS

<a href="#">1.</a>	<a href="#">Introduction.....</a>	<a href="#">2</a>
<a href="#">1.1.</a>	<a href="#">Notational Conventions.....</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Extensions to SDP and RTSP.....</a>	<a href="#">3</a>
<a href="#">2.1.</a>	<a href="#">SDP Extensions.....</a>	<a href="#">4</a>
<a href="#">2.2.</a>	<a href="#">RTSP Extensions.....</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">Usage with SIP and RTSP.....</a>	<a href="#">5</a>
<a href="#">3.1.</a>	<a href="#">General SDP processing.....</a>	<a href="#">5</a>
<a href="#">3.2.</a>	<a href="#">SIP usage.....</a>	<a href="#">6</a>
<a href="#">3.3.</a>	<a href="#">RTSP usage.....</a>	<a href="#">7</a>
<a href="#">3.4.</a>	<a href="#">Example scenarios.....</a>	<a href="#">7</a>
<a href="#">4.</a>	<a href="#">Adding further Key management protocols.....</a>	<a href="#">10</a>
<a href="#">5.</a>	<a href="#">Security Considerations.....</a>	<a href="#">10</a>
<a href="#">6.</a>	<a href="#">IANA Considerations.....</a>	<a href="#">11</a>
<a href="#">7.</a>	<a href="#">Conclusions.....</a>	<a href="#">11</a>
<a href="#">8.</a>	<a href="#">Acknowledgments.....</a>	<a href="#">11</a>
<a href="#">9.</a>	<a href="#">Author's Addresses.....</a>	<a href="#">11</a>
<a href="#">10.</a>	<a href="#">References.....</a>	<a href="#">12</a>
<a href="#">10.1.</a>	<a href="#">Normative References.....</a>	<a href="#">12</a>
<a href="#">10.2.</a>	<a href="#">Informative References.....</a>	<a href="#">12</a>

[1.](#) Introduction

There has recently been work to define a security framework for the protection of real-time applications running over RTP, [[SRTP](#)]. However, a security protocol needs a key management infrastructure to exchange keys and security parameters, managing and refreshing keys, etc.

A key management protocol is executed prior to the security protocol execution. The key management protocol's main goal is to, in a secure and reliable way, establish a so-called security association for the security protocol. This includes one or several cryptographic keys and a set of necessary parameters for the security protocol, e.g., cipher and authentication algorithm to be used. The key management protocol has similarities with, e.g., SIP [[SIP](#)] and RTSP [[RTSP](#)] in the sense that it negotiates necessary information in order to be able to setup the session.

The focus in the following sections is to describe SDP attribute extensions and RTSP header extensions to support key management, and a possible integration within SIP and RTSP. A framework is therefore described in the following. Such a framework will need to be completed by one or more key management protocols, to describe how the framework is used, e.g. which is the data to be carried in the extensions.

Some of the motivations to create a framework with the possibility to include the key management in the session establishment are:

- \* Just as the codec information is a description of how to encode and decode the audio (or video) stream, the key management data is a description of how to encrypt and decrypt the data.
- \* The possibility to negotiate the security for the entire multimedia session at the same time.
- \* The knowledge of the media at the session establishment makes it easy to tie the key management to the multimedia sessions.
- \* This approach may be more efficient than setting up the security later, as that approach might force extra roundtrips, possibly also a separate set-up for each stream, hence implying more delay to the actual setup of the media session.

Currently in SDP [[SDPnew](#)], one field exists to transport keys, i.e. the "key=" field. However, this is not enough for a key management protocol as there are many more parameters that need to be transported. The approach here is to use and extend the SDP description to transport the key management offer/answer and also to associate it with the media sessions. SIP uses the offer/answer model [[OAM](#)] whereby extensions to SDP will be enough. However, RTSP [[RTSP](#)] does not use the offer/answer model. This makes it impossible to send back an answer to the server. To solve this, a new header is introduced in which the key management data can be included.

### 1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC-2119](#).

## [2.](#) Extensions to SDP and RTSP

This section describes common attributes that are to be included in an SDP description or in an RTSP header when an integrated key management protocol is used. The attribute values MUST follow the general SDP or RTSP guidelines.

For the SDP description, the key management attributes MAY be defined at session level (i.e. before the media descriptor lines) and/or at media level. If the key management attributes are defined at media level, they will only apply to that specific media. If the key management attributes are defined at both session and media level, the media level definition overrides the session level definition for that specific media.

The following SDP attribute is defined:

key-mgmt:<identifier> <opaque-data>

<identifier> is the name of the key management protocol and the opaque-data is a field to transport the key management protocol data. The key management protocol data contains the necessary information to establish the security protocol, e.g., keys and cryptographic parameters. All parameters and keys are protected by the key management. Note that if the key management protocol fails, e.g., the receiver does not accept any of the proposed security parameters, or simply does not understand the key management protocol, the security setup will fail. Consequently, it is impossible to establish a secure session. So, if the key management fails, the offer must be rejected.

### [2.1.](#) SDP Extensions

This section provides an Augmented Backus-Naur Form (ABNF) grammar (as used in [[SDPnew](#)]) for the key management extensions to SDP.

Note that the new definitions are compliant with the definition of an attribute field, i.e.

attribute = (att-field ":" att-value) | att-field

One new attribute for SDP is defined:

key-mgmt = "key-mgmt: " prtcl-id keymgmt-data

prtcl-id = non-ws-string  
; e.g. "mikey"

keymgmt-data = text

where non-ws-string and text are as defined in SDP [[SDPnew](#)]. The attribute may be used at session level, media level or at both levels. An attribute defined at media level overrides an attribute defined at session level.

## [2.2](#). RTSP Extensions

To support the needed attribute described, the following RTSP header is defined:

KeyMgmt = "keymgmt" ":" 1#key-mgmt-spec

key-mgmt-spec = "prot" "=" token ";" "data" "=" quoted-string

token and quoted-string are as defined in the RTSP specification [[RTSP](#)].

The KeyMgmt header should be possible to use in the messages described in the table below.

Method	Direction	Requirement
DESCRIBE	C->S	required
SETUP	C->S	required
ANNOUNCE	C->S, S->C	optional (required: if re-key should be supported)

## [3](#). Usage with SIP and RTSP

This section gives recommendations of how/when to include the defined

key management attribute when SIP and/or RTSP are used together with SDP.

When a key management protocol is integrated with SIP/SDP and RTSP, the following requirements are put on the key management:

- \* It MUST be possible to execute the key management protocol in at most one roundtrip in case the answerer accepts the offer.
- \* It MUST be possible from the SIP/SDP and RTSP application, using the key management API, to receive key management data, and information of whether a message is accepted or not.

Today, the MIKEY protocol [[MIKEY](#)] has adopted the key management extensions to work together with SIP and RTSP. Other protocols MAY use the described attribute and header, e.g. Kerberos [[KERB](#)].

### [3.1](#). General SDP processing

When an SDP message is created, the following procedure should be applied:

- \* The identifier of the key management protocol used (e.g. MIKEY or Kerberos) MUST be put in the `prtcl-id` field.
- \* The `keymgmt-data` field MUST be created with the data received from the key management protocol (this data MUST be base64 encoded). The data may e.g. be a MIKEY message or Kerberos ticket.

A received SDP message that contains the key management attributes SHOULD process these attributes in the following manner:

- \* The key management protocol used MUST be identified by checking the `prtcl-id` field in the key management attribute.
- \* The key management data from the `keymgmt-data` field MUST be extracted and given to the key management protocol. Note that depending on key management protocol, some extra parameters might of course be requested, such as the source/destination network address/port(s) for the specified media.

\* Depending on the outcome of the key management processing (i.e. whether it was accepted or not), the processing can proceed according to normal processing (e.g. according to the offer/answer model, see also [Section 3.2](#)).

Note that the attribute MAY be repeated more than once (e.g., one at session level and one at media level). Consequently, the process is repeated for each key management attribute detected.

If more than one key management protocol is supported, multiple instances of the key management attribute MAY be included in the initial offer, each transporting a different key management data, thus indicating alternatives supported.

If the sender includes more than one key management protocol attributes at session level (analogous for the media level), these SHOULD be listed in order of preference (with the first being the preferred). The receiver chooses the key management protocol it supports. When answering, only the accepted key management protocol attribute MUST be included. If the receiver does not support any of the sender's suggested key management protocols, the receiver answers with an error message (see SIP and RTSP), whereby the sender MUST put down the current setup procedure.

Note that by placing multiple key management offers in a single message has the disadvantage that the message expands and the computational workload for the offerer will increase drastically. The possibility to support multiple key management protocols may introduce bidding down attacks. To avoid this, the list of identifiers of the proposed key management protocols MUST be authenticated, which MUST be done by each key management. This puts the requirement that it MUST be specified (in the key management protocol itself or in a companion document) how the protocol identifiers could be authenticated from the offerer to the responder by the use of the specific key management protocol. Note that even if only one key management protocol is used, that still must authenticate its own protocol identifier.

If more than one protocol is supported by the offerer, it is RECOMMENDED that he offers all to him acceptable protocols in the first offer, rather than making single, subsequent alternative offers in response to error messages, see "Security Considerations".

### [3.2](#). SIP usage

The offerer SHOULD include the key management data within an offer that contains the media description it should apply to. The answerer MUST check with the key management protocol if the attribute values

INTERNET-DRAFT

mmusic-kmgmt-ext-06

February 2003

are valid, and then obtain from the key management the data to include in the answer.

If the offer is not accepted, the answerer SHOULD return a "606 Not Acceptable" message, including one or more Warning headers (at least a 306). The offer MUST then abort the security setup.

Re-keying can be handled as a new offer, i.e. a re-INVITE should be sent with the new proposed parameters. The answerer treats this as a new offer where the key management is the issue of change. In general, the re-INVITE (and the key exchange) must be finalized before the security protocol can change the keys. The synchronization method used when changing keys are dependent on the security and key management protocol used.

### [3.3](#). RTSP usage

RTSP does not use the offer/answer model, as SIP does. This causes some problems, as it is not possible (without abusing RTSP) to send back an answer to the server (as the server will in most cases be the one initiating the security parameter exchange). To solve this, a new header has been introduced ([Section 2.2](#)). This also assumes that the key management also has some kind of binding to the media, so that the response to the server will be processed as required.

The processing of a key management header in RTSP should be done analogous of the SDP message processing. The initial key management message from a server should be sent to the client using SDP. When responding to this, the client uses the new RTSP header to send back an answer (included in the SETUP message). If a server receives a SETUP message in which it expects a key management message, but none is included, a 403 Forbidden SHOULD be returned to the client, whereby the current setup MUST be aborted.

The server MAY provide re-keying/updating facilities by sending a new key management message in an ANNOUNCE messages. The ANNOUNCE message contains an SDP message including the key management parameters. The response message is put in the new RTSP header in the response from the client to the server. Note that the ANNOUNCE messages MUST be supported if this feature is to be used.

### 3.4. Example scenarios

#### Example 1 (SIP)

A SIP call is taking place between Alice and Bob. Alice sends an Invite message consisting of the following offer:

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.somewhere.com
```

Arkko, et al.

[Page 7]

---

INTERNET-DRAFT

mmusic-kmgmt-ext-06

February 2003

```
s=Cool stuff
e=alice@w-land.org
t=0 0
c=IN IP4 lost.somewhere.com
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkd0okdo7eWsnDSJD...
m=audio 49000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52230 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

i.e. Alice proposes to set up one audio stream and one video stream that run over SRTP. To set up the security parameters for SRTP, she uses MIKEY. Note that MIKEY is negotiating the crypto suite for both streams (as it is placed at the session level).

Bob accepts the offer and sends an answer back to Alice:

```
v=0
o=bob 2891092897 2891092897 IN IP4 found.somewhere.com
s=Cool stuff
e=bob@null.org
t=0 0
c=IN IP4 found.somewhere.com
a=key-mgmt:mikey skaoqDeMkdwRW278HjKVB...
m=audio 49030 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52230 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

#### Example 2 (SDP)

This example shows how Alice would have done in the previous example

if she wished to protect only the audio stream.

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.somewhere.com
s=Cool stuff
e=alice@w-land.org
t=0 0
c=IN IP4 lost.somewhere.com
m=audio 49000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkd0okdo7eWsnDSJD...
m=video 52230 RTP/AVP 31
a=rtpmap:31 H261/90000
```

Note that even if the key management attribute is specified at session level, the video part will not be affected by this (as a security profile is not used).

Example 3 (RTSP)

A client wants to set up a streaming session and requests a media description from the streaming server.

```
DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
CSeq: 312
Accept: application/sdp
From: user@client.com
```

The server sends back an OK message including an SDP description.

```
RTSP/1.0 200 OK
CSeq: 312
Date: 23 Jan 1997 15:35:06 GMT
Content-Type: application/sdp
```

```
v=0
o=actionmovie 2891092738 2891092738 IN IP4 movie.somewhere.com
s=Action Movie
e=action@movie.somewhere.com
t=0 0
c=IN IP4 movie.somewhere.com
```

```
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkdOokdo7eWsnDSJD...
m=audio 0 RTP/SAVP 98
a=rtpmap:98 AMR/8000
control:rtsp://movie.somewhere.com/action/audio
m=video 0 RTP/SAVP 31
a=rtpmap:31 H261/90000
control:rtsp://movie.somewhere.com/action/video
```

The client is now ready to setup the sessions. It includes the key management data in the first message going back to the server (i.e. the SETUP message).

```
SETUP rtsp://movie.somewhere.com/action/audio RTSP/1.0
CSeq: 313
Transport: RTP/SAVP/UDP;unicast;client_port=3056-3057
keymgmt: prot=mikey; data="skaoqDeMkdwRW278HjKVB..."
```

The server processes the request including checking the validity of the key management header.

```
RTSP/1.0 200 OK
CSeq: 313
Session: 12345678
Transport: RTP/SAVP/UDP;unicast;client_port=3056-3057;
          server_port=5000-5001
```

The RTSP then proceeds as usual (with e.g. a SETUP message for the video followed by a PLAY message).

#### [4.](#) Adding further Key management protocols

This framework cannot be used with all key management protocols. The key management protocol needs to comply with the requirements described in [Section 3](#). To be able to use a key management protocol with this framework, the following MUST be specified:

- \* the key management protocol identifier that should be used in the protocol identifier fields in both SDP and RTSP (e.g. "mikey" for MIKEY).

- \* the information the key management needs from SDP and RTSP ([Section 3](#) gives a guideline of what SDP and RTSP needs from the key management). The exact API is implementation specific, but it SHOULD at least support to exchange the specified information. Note that in particular, the key management MUST always be given the protocol identifier(s) of the key management protocol(s) included in the offer in the correct order as they appear.

The key management data MUST be base64 encoded in the SDP and RTSP fields. Therefore, considerations of possible conversion from the normal key management representation to base64 SHOULD be taken into account.

## 5. Security Considerations

The nature of this document is to allow SDP and RTSP to support security of the media sessions. It is therefore not a primary intention of this document to describe possible security solutions or to define possible security problems. The defined SDP and RTSP extensions are not believed to introduce any new security risks to SDP and RTSP, if used as specified.

Note that the purpose of the key management fields is to provide information to secure the media streams. Under the assumption that the key management schemes are secure, the SDP can be passed along unprotected without affecting the key management, and the media streams will still be secure even if some attackers gained knowledge of the SDP contents.

However, if the SDP messages are not sent authenticated between the parties, it is possible for an active attacker to change attributes without being detected. As the key management protocol may (indirectly) rely on some of the session information from SDP (e.g., address information), an attack on SDP may have indirect consequences on the key management. In general, it is therefore a good thing, not only to try to secure the session, but also to secure the session setup.

offer, since even if it comes from the true respondent, the fact that he/she declines the offer usually means that he/she does not support the protocol(s) offered, and consequently cannot be expected to authenticate the response either. This means that if the initiator is unsure of which protocol(s) the responder supports, we RECOMMEND that the initiator offers all acceptable protocols in a single offer. If not, this opens up the possibility for a "man-in-the-middle" (MITM) to affect the outcome of the eventually agreed upon protocol, by faking unauthenticated error messages until the initiator eventually offers a protocol "to the liking" of the MITM. This is not really a security problem, but rather a mild form of denial of service that can be avoided by following the above recommendation.

## 6. IANA Considerations

New attribute fields for SDP (see [Section 2.1](#)) and RTSP header are registered (see [Section 2.2](#)).

## 7. Conclusions

A security solution for real-time applications needs a key management infrastructure. Integrating the key management scheme with the session establishment protocol could be done efficiently in most of the scenarios. This draft proposes a framework that integrates a key management protocol (e.g., MIKEY) into SIP and RTSP, and which can be accompanied by different key management protocols. A set of new attributes and headers has been defined in SDP and RTSP to support this.

## 8. Acknowledgments

Thanks to: Rolf Blom, Magnus Westerlund, and the rest involved in the MMUSIC WG and the MSEC WG.

A special thanks to Joerg Ott and Colin Perkins.

## 9. Author's Addresses

Jari Arkko

Ericsson

02420 Jorvas

Finland

Phone: +358 40 5079256

Email: [jari.arkko@ericsson.com](mailto:jari.arkko@ericsson.com)

Elisabetta Carrara

Ericsson Research

INTERNET-DRAFT

mmusic-kmgmt-ext-06

February 2003

SE-16480 Stockholm      Phone: +46 8 50877040  
Sweden                    EMail: elisabetta.carrara@era.ericsson.se

Fredrik Lindholm  
Ericsson Research  
SE-16480 Stockholm      Phone: +46 8 58531705  
Sweden                    EMail: fredrik.lindholm@era.ericsson.se

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

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

## 10. References

### 10.1. Normative References

[OAM] Rosenberg, J. and Schulzrinne, H., "An Offer/Answer Model with the Session Description Protocol (SDP)", IETF, 3264.

[RTSP] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time Streaming Protocol (RTSP)", IETF, [RFC 2326](#).

[SDPnew] Handley, M., Jacobson, V., and Perkins, C., "SDP: Session Description Protocol", Internet Draft, IETF, Work in progress (MMUSIC).

[SIP] Handley, M., Schulzrinne, H., Schooler, E., and Rosenberg, J., "SIP: Session Initiation Protocol", IETF, [RFC 2543](#).

### 10.2. Informative References

[KERB] Kohl, J., Neuman, C., "The Kerberos Network Authentication Service (V5)", IETF, [RFC 1510](#).

[MIKEY] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and

Norrman, K., "MIKEY: Multimedia Internet KEYing", Internet Draft, IETF, Work in progress (MSEC).

[SRTP] Baugher, M., Blom, R., Carrara, E., McGrew, D., Naslund, M, Norrman, K., and Oran, D., "The Secure Real Time Transport Protocol", Internet Draft, IETF, Work in Progress (AVT).

This Internet-Draft expires in August 2003.