Internet Engineering Task Force
MMUSIC Working Group
INTERNET-DRAFT
Expires: February 2004

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

August 2003

Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)
<draft-ietf-mmusic-kmgmt-ext-08.txt>

Status of this memo

Copyright Notice

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.

TABLE OF CONTENTS

1. Introduction

[Editor remark] All instances of RFC xxxx should be replaced with the RFC number of this document, when published. Furthermore, all instances of RFC yyyy should be replaced with the RFC number of the MIKEY (Multimedia Internet KEYing) document [MIKEY], when published.

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, manage and refresh 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 security association for the security protocol. This includes one or more cryptographic keys and the 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 a new SDP attribute and RTSP header extension to support key management, and the integration within SIP and RTSP. A framework is therefore described in the following. This framework is 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.

* The possibility to negotiate keying material end-to-end without applying end-to-end protection of the SDP (instead, hop-by-hop security mechanisms can be used which may be useful if intermediate proxies needs access to the SDP).

Currently in SDP [SDPnew], one field exists to transport keys, i.e. the "k=" 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 with SDP, so a new header is introduced to convey key management data.

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 [RFC2119].

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 (see [SDPnew] and [RTSP]).

For both SDP and RTSP, the general method of adding the key management protocol is to introduce new attributes, one identifier to identify the specific key management protocol, and one data field where the key management protocol data is placed. 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.

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. Note that the prtcl-id name will be case sensitive and it is therefore RECOMMENDED that attributes registered are in lower case letters. Section 3 describes in detail how the attributes are used and how the SDP is handled in different usage scenarios.

2.2. RTSP Extensions

To support the needed attribute, 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 is supported)

Note: Section 3 describes in detail how the RTSP extensions are used.

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 placed 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 placed in the prtcl-id field.

* The keymgmt-data field MUST be created as follows. The key management protocol MUST be used to create the key management message. This message SHALL be base64 encoded [RFC3548] by the SDP application and then encapsulated in the keymgmt-data attribute. The data may e.g. be a MIKEY message (see [MIKEY], Section 7) or Kerberos ticket.

A received SDP message that contains the key management attributes SHOULD be processed in the following manner:

* The key management protocol is identified according to the prtcl-id field.

* The key management data from the keymgmt-data field MUST be extracted, base64 decoded to reconstruct the original message, and then passed to the key management protocol. Note that depending on key management protocol, some extra parameters might of course be requested by the specific API, such as the source/destination network address/port(s) for the specified media (however, this will be implementation specific depending on the actual API).

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

Note that the key management 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. However, in case of failure of the key management (on either session or media level), the session setup SHALL be aborted (see also Section 3.2 and Section 3.3 for more details).

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 supported alternatives.

If the sender includes more than one key management protocol attribute at session level (analogous for the media level), these SHOULD be listed in order of preference (the first being the preferred). The receiver selects the key management protocol it wishes to use and includes only that attribute in the answer. If the receiver does not support any of the sender's suggested key management protocols, the receiver returns an error message (see section 3.2 and section 3.3), whereby the sender MUST abort the current setup procedure.

Note that the placement of 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. The authentication MUST be done separately by each key management protocol (see e.g. Section 7.1 in [MIKEY]).

Accordingly, it MUST be specified (in the key management protocol specification itself or in a companion document) how the list of key management protocol identifiers can be authenticated from the offerer to the answerer by the specific key management protocol. Note that even if only one key management protocol is used, that still MUST authenticate its own protocol identifier.

The list of protocol identifiers MUST be given to the selected key management protocol by the SDP application with ";" separated identifiers. All the offered protocol identifiers MUST be included, in the same order as they appear in the corresponding SDP description.

The protocol list can formally be described as

```
prtcl-list   =  prtcl-id *(";" prtcl-id)

prtcl-id     = non-ws-string
```

For example, if the SDP is:

```
v=0
o=alice 2891092738 2891092738 IN IP4 lost.example.com
s=Secret discussion
t=0 0
c=IN IP4 lost.example.com
a=key-mgmt:mikey <data1>
a=key-mgmt:keyp1 <data2>
a=key-mgmt:keyp2 <data3>
m=audio 39000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 42000 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

The protocol list, "mikey;keyp1;keyp2", would be generated from the SDP description and used as input to each specified key management protocol (together with the data for that protocol).

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

3.2. SIP usage

When used with SIP and the offer/answer model, 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 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 "Attribute not understood"). The session is then aborted (and it is up to local policy or end user to decide how to continue).

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 same key management protocol used in the original INVITE SHALL also be used in the re-INVITE carrying re-keying. If the re-INVITE carrying re-keying fails (e.g., the authentication verification fails), the answerer SHOULD send a "606 Not Acceptable" message, including one or more Warning headers (at least a 306). The offer MUST then abort the security setup.

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 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 processing of creating a key management header in RTSP SHOULD be as follow:

* The identifier of the key management protocol used (e.g. MIKEY or Kerberos) MUST be placed in the "prot" field of the header.

* The keymgmt-data field MUST be created as follows. The key management protocol MUST be used to create the key management message. This message SHALL be base64 encoded by the SDP application and then encapsulated in the "data" field of the header. The data may e.g. be a MIKEY message (see [MIKEY], Section 7) or Kerberos ticket.

A received key management header SHOULD be processed in the following manner:

- The key management protocol is identified according to the "prot" field.

- The key management data from the "data" field MUST be extracted, base64 decoded to reconstruct the original message, and then passed to the key management protocol. Note that depending on the key management protocol, some extra parameters might of course be requested by the specific API, such as the source/destination network address/port(s) for the specified media (however, this will be implementation specific depending on the actual API).

- Depending on the outcome of the key management processing (i.e. whether it was successful or not), the processing can proceed according to normal rules (see also below).

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 w-land.example.com
s=Cool stuff
e=alice@w-land.example.com
t=0 0
c=IN IP4 w-land.example.com
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkdOokdo7eWsnDSJD...
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 foo.example.com
s=Cool stuff
e=bob@foo.example.com
t=0 0
c=IN IP4 foo.example.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 if she wished to protect only the audio stream.

```
v=0
o=alice 2891092738 2891092738 IN IP4 w-land.example.com
s=Cool stuff
e=alice@w-land.example.com
t=0 0
c=IN IP4 w-land.example.com
m=audio 49000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkdOokdo7eWsnDSJD...
m=video 52230 RTP/AVP 31
a=rtpmap:31 H261/90000
```

Note that even if the key management attribute were specified at session level, the video part would 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@example.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.example.com
s=Action Movie
e=action@movie.example.com
t=0 0
c=IN IP4 movie.example.com
a=key-mgmt:mikey uiSDF9sdhs727ghsd/dhsoKkdOokdo7eWsnDSJD...
m=audio 0 RTP/SAVP 98
a=rtpmap:98 AMR/8000
control:rtsp://movie.example.com/action/audio
m=video 0 RTP/SAVP 31
a=rtpmap:31 H261/90000
control:rtsp://movie.example.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.example.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 negotiation of the 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. Even if the key management protocol does not rely on parameters of SDP and will not be affected by manipulation of these, different DoS attacks aimed at SDP (e.g. the SIMCAP extensions) may lead to undesired interruption in the setup.

In general, it is therefore a good thing, not only to try to secure the session, but also to secure the session setup. However, the security of the session setup might not possible on an end-to-end basis, but may require to be protected on a hop-by-hop basis (this is generally the case for SIP/SDP when intermediate proxies needs to obtain information about the sessions etc). In fact, the focus of this framework is mainly when end-to-end protection of the session setup is not used, but where the media streams needs to be end-to-end protected.

Note that it is impossible to assure the authenticity of a declined offer, since even if it comes from the true respondent, the fact that the answerer declines the offer usually means that he 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

6.1. SDP Attribute Registration

A new SDP attribute needs to be registered for the purpose of key management protocol integration with SDP.

Contact: Fredrik Lindholm
mailto: fredrik.lindholm@ericsson.com
tel: +46 8 58531705

SDP Attribute ("att-field"):

Name: key-mgmt
Long form: key management protocol
Type of name: att-field
Type of attribute: Media and session level
Purpose: See RFC xxxx, Section 2.
Reference: RFC xxxx, Section 2.1
Values: See registrations below

6.2. Protocol Identifier Registration

This document defines one new name space associated with the above registered key-mgmt attribute i.e., the protocol identifier (see also Section 2.1 and Section 2.2).

A new registry needs to be set up for "prtcl-id" parameter of the "key-mgmt" attribute, with the following registration created initially: "mikey".

Contact: Fredrik Lindholm
mailto: fredrik.lindholm@ericsson.com
tel: +46 8 58531705

Value name: mikey
Long name: Multimedia Internet KEYing
Purpose: Usage of MIKEY with the key-mgmt attribute

Reference: Section 7 in RFC yyyy

Further entries may be registered according to the "Specification Required" policy as defined in [RFC2434]. Each new registration needs to indicate the parameter name and the syntax of possible additional arguments. Note that the parameter name is case sensitive and it is recommended that the name should be in lower case letters. For each new registration, it is mandatory that a permanent, stable, and publicly accessible document exists that specifies the semantics of the registered parameter, the syntax and semantics of its parameters as well as all the requested details of interaction between the key management protocol and SDP, as specified in this document.

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 Phone: +358 40 5079256
Finland Email: jari.arkko@ericsson.com

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

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

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

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

10. References

10.1. Normative References

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

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

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

[SDPnew] Handley, M., Jacobson, V., and Perkins, C., "SDP: Session Description Protocol", Internet Draft, IETF, Work in progress (MMUSIC), draft-ietf-mmusic-sdp-new-13.txt.

[SIP] Handley, M., Schulzrinne, H., Schooler, E., and Rosenberg, J., "SIP: Session Initiation Protocol", IETF, RFC 3261.

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

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

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", IETF, RFC yyyy, [Internet Draft, 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).

Copyright Notice

This Internet-Draft expires in February 2004.