Internet Engineering Task Force
MMUSIC Working Group
INTERNET-DRAFT
Expires: December 2005

J. Arkko
E. Carrara
F. Lindholm
M. Naslund
K. Norrman
Ericsson
June 2005

Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)

<draft-ietf-mmusic-kmgmt-ext-15.txt>

Status of this memo

Copyright Notice

Abstract

This document defines general extensions for SDP and RTSP to carry messages, as specified 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, their use

is meaningful only when complemented by an appropriate key management protocol.

General guidelines are also given on how the framework should be used together with SIP and RTSP. The usage with the MIKEY key management protocol is also defined.

TABLE OF CONTENTS

1. Introduction

[RFC Editor remark] All instances of RFC xxxx should be replaced with the RFC number of this document, when published.

There has recently been work to define a security profile for the protection of real-time applications running over RTP, [SRTP]. However, a security protocol needs a key management solution to exchange keys and security parameters, manage and refresh keys, etc.

A key management protocol is executed prior to the security protocol's 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 algorithms 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 to show how these can be integrated within SIP and RTSP. The resulting framework is completed by one or more key management protocols, which use the extensions provided.

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 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], there exists one field to transport keys, the "k=" field. However, this is not enough for a key management protocol as there are many more parameters that need to be transported, and the "k=" field is not extensible. The approach used is to extend the SDP description through a number of attributes that 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 RTSP header is introduced to convey key management data. [SDES] uses the approach of extending SDP, to carry the security parameters for the media streams. However, the mechanism defined in [SDES] requires end-to-end

protection of the SDP by some security protocol such as S/MIME, in order to get end-to-end protection. The solution described here

focuses only on the end-to-end protection of key management parameters and as a consequence does not require external end-to-end protection means. It is important to note though, and we stress this again, that only the key management parameters are protected.

The document also defines the use of the described framework together with the key management protocol Multimedia Internet KEYing (MIKEY) [MIKEY].

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. Applicability

[SDES] provides similar cryptographic key distribution capabilities, and it intended for use when keying material is protected along with the signaling.

In contrast, this specification expects endpoints to have preconfigured keys or common security infrastructure. It provides its own security, and is independent of the protection of signaling (if any). As a result, it can be applied in environments where signaling protection is not turned on, or used hop-by-hop (i.e., scenarios where the SDP is not protected end-to-end). This specification will independently of the signaling protection applied, ensure end-to-end security establishment for the media.

3. Extensions to SDP and RTSP

This section describes common attributes that can be included in SDP or RTSP when an integrated key management protocol is used. The attribute values follow the general SDP and 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 protocol.

The key management data SHALL be base64 [RFC3548] encoded and comply with the base64 grammar as defined in [SDPnew]. The key management

protocol identifier, KMPID, is defined as below in Augmented Backus-Naur Form grammar (ABNF) [RFC2234].

```
KMPID =  1*(ALPHA / DIGIT)
```

Values for the identifier, KMPID, are registered and defined in accordance to Section 8. Note that the KMPID is case sensitive and it is RECOMMENDED that values registered are lower case letters.

3.1. SDP Extensions

This section provides an 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
```

The ABNF for the key management extensions (conforming to the att-field and att-value) are as follow:

```
  key-mgmt-attribute = key-mgmt-att-field ":" key-mgmt-att-value

  key-mgmt-att-field = "key-mgmt"
  key-mgmt-att-value = 0*1SP prtcl-id SP keymgmt-data

  prtcl-id     = KMPID
                 ; e.g. "mikey"

  keymgmt-data = base64
  SP           = 0x20
```

where KMPID is as defined in Section 2 of this memo, base64 is as defined in SDP [SDPnew]. Prtcl-id refers to the set of values defined for KMPID in Section 8.

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. In other words, if the media level attribute is present, the session level attribute MUST be ignored for this media. Section 3.1 describes in detail how the attributes are used and how the SDP is handled in different usage scenarios. The choice of the level depends for example on the particular key management protocol. Some protocols may not be able to derive enough key material for all the sessions; furthermore, possibly a different protection to each session could be required. The particular protocol might achieve this only by specifying it at the media level. Other protocols, such as MIKEY, have instead those capabilities (as it can express multiple security policies and derive multiple keys), so it may use the session level.

3.2. RTSP Extensions

To support the key management attributes, the following RTSP header is defined:

KeyMgmt = "KeyMgmt" ":" key-mgmt-spec 0*("," key-mgmt-spec)

key-mgmt-spec = "prot" "=" KMPID ";" ["uri" "=" <"> rtsp_URL <"> ";"] "data" "=" base64

where KMPID is as defined in Section 2 of this memo, "base64" as defined in [SDPnew], and "rtsp_URL" as defined in [RTSP].

The "uri" parameter identifies the context for which the key management data applies, and the RTSP URI SHALL match a (session or media) URI present in the description of the session. If the RTSP aggregated control URI is included it indicates that the key management message is on session level (and similarly the RTSP media control URI, that it applies to the media level). If no "uri" parameter is present in a key-mgmt-spec the specification applies to the context identified by the RTSP request URI.

The KeyMgmt header MAY be used in the messages and directions described in the table below.

Method	Direction	Requirement
DESCRIBE response	S->C	RECOMMENDED
SETUP	C->S	REQUIRED
SETUP Response	S->C	REQUIRED (error)

Note: Section 3.2 describes in detail how the RTSP extensions are used.

We define one new RTSP status code to report error due to any failure during the key management processing (Section 3.2):

Status-Code = "463" ; Key management failure

A 463 response MAY contain a KeyMgmt header with a key management protocol message that further indicates the nature of the error.

4. Usage with SDP, SIP, RTSP, and SAP

This section gives rules and 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 general requirements are placed on the key management:

* At the current time, it MUST be possible to execute the key management protocol in at most one request-response message exchange. Future relaxation of this requirement is possible but would introduce significant complexity for implementations supporting multi-roundtrip mechanisms.

* 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.

The content of the key management messages depends on the key management protocol that is used. However, the content of such key management messages might be expected to be roughly as follow: the key management Initiator (e.g. the offerer) includes the key management data in a first message, containing the media description it should apply to. This data in general consists of the security parameters (including key material) needed to secure the communication, together with the necessary authentication information (to assure that the message is authentic).

At the Responder's side, the key management protocol checks the validity of the key management message, together with the availability of the parameters offered, and then provides the key management data to be included in the answer. This answer may typically authenticate the Responder to the Initiator, and also state if the initial offer was accepted or not. Certain protocols might require the Responder to include a selection of the security parameters that he is willing to support. Again, the actual content of such responses is dependent on the particular key management protocol.

Section 6 describes a realization of the MIKEY protocol using these mechanisms. Procedures to be used when mapping new key management protocols onto this framework are described in Section 5.

4.1. Use of SDP

This section describes the processing rules for the different applications which use SDP for the key management.

4.1.1 General processing

The processing when SDP is used is slightly different according to the way SDP is transported, and if it uses an offer/answer or announcement. The processing can be divided into four different steps:

1) How to create the initial offer.

2) How to handle a received offer.
3) How to create an answer.
4) How to handle a received answer.

It should be noted that the last two steps may not always be applicable, as there are cases where an answer can not or will not be sent back.

The general processing for creating an initial offer SHALL follow the following actions:

* The identifier of the key management protocol used MUST be placed in the prtcl-id field of SDP. A table of legal protocols identifiers is maintained by IANA (see Section 8).

* 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. Note though that the semantics of the encapsulated message is dependent on the key management protocol that is used.

The general processing for handling a received offer SHALL follow the following actions:

* The key management protocol is identified according to the prtcl-id field. A table of legal protocols identifiers is maintained by IANA (Section 8).

* 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 for processing. Note that depending on key management protocol, some extra parameters might also 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). The extra parameters that a key management protocol might need (other than the ones defined here) MUST be documented, describing their use, as well as the interaction of that key management protocol with SDP and RTSP.

* If errors occur, or the key management offer is rejected, the session SHALL be aborted. Possible error messages are dependent on the specific session establishment protocol.

At this stage, the key management will have either accepted or rejected the offered parameters. This MAY cause a response message to be generated, depending on the key management protocol and the application scenario.

If an answer is to be generated, the following general actions SHALL be performed:

* The identifier of the key management protocol used 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 semantics of the encapsulated message is dependent on the key management protocol that is used.

The general processing for handling a received answer SHALL follow the following actions:

* 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 for processing.

* If the key management offer is rejected and the intent is to re-negotiate it, it MUST be done through another Offer/Answer exchange. It is RECOMMENDED to NOT abort the session in that case, but to re-negotiate using another Offer/Answer exchange. For example, in SIP [RFC3261], the "security precondition" as defined in [SPREC} solves the problem for a session initiation. The procedures in [SPREC] are outside the scope of this document. In an established session, an additional Offer/Answer exchange using a re-INVITE or UPDATE as appropriate MAY be used.

* If errors occur, or the key management offer is rejected and there is no intent to re-negotiate it, the session SHALL be aborted. If possible an error message indicating the failure SHOULD be sent back.

Otherwise, if all the steps are successful, the normal setup proceeds.

4.1.2 Use of SDP with offer/answer and SIP

This section defines additional processing rules, to the general rules defined in Section 3.1.1, applicable only to applications using SDP with the offer-answer model [OAM] (and in particular SIP).

When an initial offer is created, the following offer-answer specific procedure SHALL be applied:

* Before creating the key management data field, the list of protocol identifiers MUST be provided by the SDP application to (each) key management protocol, as defined in Section 3.1.4 (to defeat bidding-down attacks).

For a received SDP offer that contains the key management attributes, the following offer-answer specific procedure SHALL be applied:

* Before, or in conjunction with, passing the key management data to the key management protocol, the complete list of protocol identifier from the offer message is provided by the SDP application to the key management protocol (as defined in Section 3.1.4).

When an answer is created, the following offer-answer specific procedure SHALL be applied:

* If the key management rejects the offer and the intent is to re-negotiate it, the Answer SHOULD include the cause of failure in an included message from the key management protocol. The renegotiation MUST be done through another Offer/Answer exchange (e.g, using [SPREC]). In an established session, it can also be done through a re-INVITE or UPDATE as appropriate.

* If the key management rejects the offer and the session needs to be aborted, the answerer SHOULD return a "488 Not Acceptable Here" message, optionally also including one or more Warning headers (a 306 "Attribute not understood" when one of the parameters is not supported, and a 399 "Miscellaneous warning" with arbitrary information to be presented to a human user or logged, see Section 20.43 in [SIP]). Further details about the cause of failure MAY be described in an included message from the key management protocol. The session is then aborted (and it is up to local policy or end user to decide how to continue).

Note that the key management attribute (related to the same key management protocol) MAY be present both at session level and at media level. Consequently, the process SHALL be repeated for each such key management attribute detected. In case the key management processing of any such attribute does not succeed (e.g. authentication failure, parameters not supported etc.), on either session or media level, the entire session setup SHALL be aborted, including those parts of the session that successfully completed their part of the key management.

If more than one key management protocol is supported, multiple instances of the key management attribute MAY be included in the initial offer when using the offer-answer model, each transporting a

different key management protocol, thus indicating supported alternatives.

If the offerer 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 answerer selects the key management protocol it wishes to use, and processes only it, on either session or media level, or on both, according to where located. If the answerer does not support any of the offerer's suggested key management protocols, the answerer indicates this to the offerer so a new Offer-Answer can be triggered; alternatevely, it may return a "488 Not Acceptable Here" error message, 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. Unless the guidelines of Section 3.1.4 are followed, multiple lines may open up bidding-down attacks. Note also that the multiple offer option has been added to optimize signaling overhead in case the Initiator knows some key (e.g. a public key) that the Responder has, but is unsure of what protocol the Responder supports. The mechanism is not intended to negotiate options within one and the same protocol.

The offerer MUST include the key management data within an offer that contains the media description it applies to.

Re-keying MUST be handled as a new offer, with the new proposed parameters. The answerer treats this as a new offer where the key management is the issue of change. The re-keying exchange MUST be finalized before the security protocol can change the keys. The same key management protocol used in the original offer SHALL also be used in the new offer carrying re-keying. If the new offer carrying re-keying fails (e.g., the authentication verification fails), the answerer SHOULD send a "488 Not Acceptable Here" message, including one or more Warning headers (at least a 306). The offerer MUST then abort the session.

Note that, in multicast scenarios, unlike unicast, there is only a single view of the stream [OAM], hence there MUST be a uniform agreement of the security parameters.

After the offer is issued, the offerer SHOULD be prepared to receive media, as the media may arrive prior to the answer. However, this brings issues, as the offer does not know yet the answererÆs choice in terms of e.g. algorithms, nor possibly the key is know. This can cause delay or clipping can occur; if this is unacceptable, the offerer SHOULD use mechanisms outside the scope of this document, e.g. the security precondition for SIP [SPREC].

4.1.3 Use of SDP with SAP

There are cases where SDP is used without conforming to the offer/answer model; instead it is a one-way SDP distribution (i.e. without back channel), such as when used with SAP and HTTP.

The processing follows the two first steps of the general SDP processing (see Section 3.1.1). It can be noted that the processing in this case differs from the offer/answer case in the fact that only one key management protocol SHALL be offered (i.e. no negotiation will be possible). This implies that the bidding down attack is not an issue; therefore the countermeasure is not needed. The key management protocol used MUST support one-way messages.

4.1.4 Bidding-down attack prevention

The possibility to support multiple key management protocols may, unless properly handled, introduce bidding-down attacks. Specifically, a man-in-the-middle could "peel off" cryptographically strong offers (deleting the key management lines from the message), leaving only weaker ones as the Responder's choice. 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.

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 processed to 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 then be given to each of the selected (offered) key management protocols by the 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   =  KMPID *(";" KMPID)
```

where KMPID is as defined in Section 2.

For example, if the offered protocols are MIKEY and two yet-to-be-invented protocols KEYP1, KEYP2, 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 AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=key-mgmt:keyp1 727gkdOshsuiSDF9sdhsdKnD/dhsoSJokdo7eWD...
a=key-mgmt:keyp2 DFsnuiSDSh9sdh Kksd/dhsoddo7eOok727gWsJD...
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). Each of the three protocols includes this protocol identifier list in its authentication coverage (according to its protocol specification).

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".

End-to-end integrity protection of the key-mgmt attributes altogether, provided externally to the key management themselves, also gives protection against this bidding down attack. This is for example the case if SIP uses S/MIME [RFC3851] to end-to-end integrity protect the SDP description. As however this end-to-end protection is not an assumption of the framework, the mechanisms defined in this section SHALL be applied.

4.2. RTSP usage

RTSP does not use the offer/answer model, as SIP does. This causes some problems, as it is not possible (without modifying RTSP) to send back an answer. 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 server SHALL be the Initiator of the key management exchange for sessions in PLAY mode, i.e. transporting media from server to client. The below text describes the behavior for PLAY mode. For any other mode the behavior is not defined in this specification.

To obtain a session description, the client initially contacts the server via a DESCRIBE message. The initial key management message from the RTSP server is sent to the client in the SDP of the 200 OK in response to the DESCRIBE. Note that only one key management protocol SHALL be used per session / media level. A server MAY allow

the SDP with key-management attribute(s) to be distributed to the client though other means than RTSP, although this is not specified here.

The "uri" parameter of the KeyMgmt header is used to indicate for the key management protocol on what context the carried message applies. For key management messages on the SDP session level, the answer MUST contain the RTSP aggregated control URL to indicate this. For Key management messages initially on SDP media level, the key management response message in the KeyMgmt header MAY use the RTSP media level URL. For RTSP sessions not using aggregated control, i.e. no session level control URI is defined, the key management protocol SHALL only be invoked on individual media streams. In this case also, the key management response SHALL be on individual media streams (i.e. one RTSP key management header per media).

When responding to the initial key management message, the client uses the new RTSP header (KeyMgmt) to send back an answer. How this is done depends on the usage context:

* Key management protocol responses for the initial establishment of security parameters for an aggregated RTSP session SHALL be sent in the first SETUP of the session. This means that if the key management is declared for the whole session but is setup in non-aggregated fashion, i.e. one media per RTSP session, each SETUP MUST carry the same response for the session level context. When performing a setup of the second or any subsequent media in a RTSP session the same key management parameters as established for the first media also applies to these setups.

* Key management responses for the initial establishment of security parameters for an individual media SHALL only be included in SETUP for the corresponding media stream.

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.

When the server creates an initial SDP message, the procedure SHALL be the same as described in Section 3.1.1.

The client processing of the initial SDP message from the server SHALL follow the same procedures as described in Section 3.1.1, except that, if there is an error, the session is aborted (no error is sent back).

The client SHALL create the response, using the key management header in RTSP, as follows:

* The identifier of the key management protocol used (e.g. MIKEY) MUST be placed in the "prot" field of the header. The prot values are maintained by IANA (Section 8).

* 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 RTSP application and then encapsulated in the "data" field of the header. The semantic of the encapsulated message is dependent on the key management protocol that is used.

* Include, if necessary, the URL to indicate the context in the "uri" parameter.

The server SHALL process a received key management header in RTSP as follow:

* 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 for processing.

* If the key management protocol is successful, the processing can proceed according to normal rules.

* Otherwise, if the key management fails (e.g. due to authentication failure or parameter not supported), an error is sent back as the SETUP response using RTSP error code 463 (see Section 2.2) and the session is aborted. It is up to the key management protocol to specify (within the RTSP status code message or through key management messages) details about the type of error that occurred.

Re-keying within RTSP is for further study, given that media updating mechanisms within RTSP are unspecified at the time this document is written.

5. Example scenarios

The following examples utilize MIKEY [MIKEY] as the key management protocol to be integrated into SDP and RTSP (see Section 5.1.).

Example 1 (SIP/SDP)

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 AQAFgM0XflABAAAAAAAAAAAAAAsAyONQ6gAAAAAGEEoo2pee4hp2
UaDX8ZE22YwKAAAPZG9uYWxkQGR1Y2suY29tAQAAAAAAAQAk0JKpgaVkDaawi9whVBtBt
0KZ14ymNuu62+Nv3ozPLygwK/GbAV9iemnGUIZ19fWQUOSrzKTAv9zV
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 (signaled by the use of the SAVP profile). She uses MIKEY to set up the security parameters for SRTP (Section 6). The MIKEY message contains the security parameters, together with the necessary key material. Note that MIKEY is exchanging the crypto suite for both streams, as it is placed at the session level. Also, MIKEY provides its own security, i.e. when Bob processes Alice's MIKEY message, he will also find the signaling of the security parameters used to secure the MIKEY exchange. Alice's endpoint's authentication information is also carried within the MIKEY message, to prove that the message is authentic. The above MIKEY message is an example of message when the pre-shared method MIKEY is used.

Upon receiving the offer, Bob checks the validity of the received MIKEY message, and, in case of successful verification, he accepts the offer and sends an answer back to Alice (with his authentication information, and, if necessary, also some key material from his side):

```
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 AQEFgM0XflABAAAAAAAAAAAAAAYAyONQ6gAAAAAJAAAQbWlja2
V5QG1vdXNlLmNvbQABn8HdGE5BMDXFIuGEga+62AgY5cc=
m=audio 49030 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52230 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

Upon receiving the answer, Alice verifies the correctness of it. In case of success, at this point Alice and Bob share the security parameters and the keys needed for a secure RTP communication.

Example 2 (SDP)

This example shows how Alice would have done if she wished to protect only the audio stream. She would have placed the MIKEY line at media level for the audio stream only (also specifying the use of the SRTP

profile there, SAVP). The semantic of the MIKEY messages is as in the previous case, but applies only to 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 AQAFgM0XflABAAAAAAAAAAAAAAsAy...
m=video 52230 RTP/AVP 31
a=rtpmap:31 H261/90000
```

Bob would then act as described in the previous example, including the MIKEY answer at the media level for the audio stream (as Alice did).

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, instead the RTP/AVP profile is signaled).

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, together with the MIKEY message. The MIKEY message contains the necessary security parameters that the server is willing of offering to the client, together with authentication information (to prove that the message is authentic) and the key material. The SAVP profile also signals the use of SRTP for securing the media sessions.

```
RTSP/1.0 200 OK
CSeq: 312
Date: 23 Jan 1997 15:35:06 GMT
Content-Type: application/sdp
Content-Length: 478

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=control:rtsp://movie.example.com/action
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAy...
m=audio 0 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=control:rtsp://movie.example.com/action/audio
m=video 0 RTP/SAVP 31
a=rtpmap:31 H261/90000
a=control:rtsp://movie.example.com/action/video
```

The client checks the validity of the received MIKEY message, and, in case of successful verification, it accept the message. The client then includes its key management data in the SETUP request going back to the server, the client authentication information (to prove that the message is authentic) and, if necessary, some key material.

```
SETUP rtsp://movie.example.com/action/audio RTSP/1.0
CSeq: 313
Transport: RTP/SAVP/UDP;unicast;client_port=3056-3057
keymgmt: prot=mikey; uri="rtsp://movie.example.com/action";
         data="AQEFgM0XflABAAAAAAAAAAAAAAYAyONQ6g..."
```

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
```

Note than in this case the key management line was specified at the session level, the key management information only goes into the SETUP related to the first stream. The "uri" indicates to the server that the context is for the whole aggregated session the key management applies. The RTSP client then proceeds setting up the second media (video) in aggregation with the audio. As the two media are run in aggregation and the key context was established in the first exchange, no more key management messages are needed.

Example 4 (RTSP)

The use of the MIKEY message at the media level would change the previous example as follows.

The 200 OK would contain the two distinct SDP attributes for MIKEY at

the media level:

```
RTSP/1.0 200 OK
CSeq: 312
Date: 23 Jan 1997 15:35:06 GMT
Content-Type: application/sdp
Content-Length: 561

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=control:rtsp://movie.example.com/action
m=audio 0 RTP/SAVP 98
a=rtpmap:98 AMR/8000
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAA...
a=control:rtsp://movie.example.com/action/audio
m=video 0 RTP/SAVP 31
a=rtpmap:31 H261/90000
a=key-mgmt:mikey AQAFgM0AdlABAAAAAAAAAAAA...
a=control:rtsp://movie.example.com/action/video
```

Each RTSP header are inserted in the SETUP related to the audio and video separately:

```
SETUP rtsp://movie.example.com/action/audio RTSP/1.0
CSeq: 313
Transport: RTP/SAVP/UDP;unicast;client_port=3056-3057
keymgmt: prot=mikey; uri="rtsp://movie.example.com/action/audio";
         data="AQEFgM0XflABAAAAAAAAAAAA..."
```

and similarly for the video session:

```
SETUP rtsp://movie.example.com/action/video RTSP/1.0
CSeq: 315
Transport: RTP/SAVP/UDP;unicast;client_port=3058-3059
keymgmt: prot=mikey; uri="rtsp://movie.example.com/action/video";
         data="AQEFgM0AdlABAAAAAAAAAAAA..."
```

Note: The "uri" parameter could be excluded from the two SETUP messages in this example.

6. 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. In addition to this, the following needs to

be defined:

* The key management protocol identifier to be used as the protocol identifier should be registered at IANA according to Section 8.

* The information that the key management needs from SDP and RTSP, and vice versa, as described in Section 3. The exact API is implementation specific, but it MUST at least support the exchange of the specified information.

* The key management protocol to be added MUST be such, that the processing in Section 3 (describing its interactions with SDP and RTSP) can be applied. Note in particular, Section 3.1.4 requires each key management protocol to specify how the list of protocol identifiers is authenticated inside that key management protocol. 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.

Finally, it is obviously crucial to analyze possible security implications induced by the introduction of a new key management protocol in the described framework.

Today, the MIKEY protocol [MIKEY] has adopted the key management extensions to work together with SIP and RTSP (see Section 6). Other protocols MAY use the described attribute and header, e.g. Kerberos [KERB], however this is subject to future standardization.

7. Integration of MIKEY

[MIKEY] describes a key management protocol for real-time applications (both for peer-to-peer communication and group communication). MIKEY carries the security parameters needed for setting up the security protocol (e.g., SRTP) protecting the media stream. MIKEY can be integrated within SDP and RTSP, following the rules and guidelines described in this document.

MIKEY satisfies the requirements described in Section 3. The MIKEY message is formed as defined in [MIKEY], then passed from MIKEY to the SDP application that base64 encodes it, and encapsulates it in the keymgmt-data attribute. The examples in Section 4 use MIKEY, where the semantic of the exchange is also briefly explained.

The key management protocol identifier (KMPID) to be used as the protocol identifier SHALL be "mikey" and is registered at IANA, see in detail Section 8.

The information that the key management needs from SDP and RTSP, and vice versa, follows Section 3. To avoid bidding-down attacks, the directives in Section 3.1.4 are followed. The list of protocol

identifiers is authenticated within MIKEY by placing the list in a General Extension Payload (of type "SDP IDs", [MIKEY]), which then

automatically will be integrity protected/signed. The receiver SHALL then match the list in the General Extension Payload with the list included in SDP and SHOULD (according to policy) if they differ, or if integrity/signature verification fails, reject the offer.

The server will need to be able to know the identity of the Client before creating and sending a MIKEY message. To signal the (MIKEY) identity of the client to the server in the DESCRIBE, it is RECOMMENDED to include the From header field in RTSP. Other methods to establish the identity could be using the IP address or retrieving the identity from the RTSP authentication if used.

7.1 MIKEY Interface

This subsection describes some aspects, which implementers SHOULD consider. If the MIKEY implementation is separate from the SDP/SIP/RTSP, an application programming interface (API) between MIKEY and those protocols is needed with certain functionality (however, exactly what it looks like is implementation dependent).

The following aspects need to be considered:

* the possibility for MIKEY to receive information about the sessions negotiated. This is to some extent implementation dependent. But it is RECOMMENDED that, in the case of SRTP streams, the number of SRTP streams is included (and the direction of these). It is also RECOMMENDED to provide the destination addresses and ports to MIKEY. When referring to streams described in SDP, MIKEY SHALL allocate two consecutive numbers for the related Crypto Session indexes (as each stream can be bi-directional). An example: if the SDP contains two m lines (specifying whatever direction of the streams), and MIKEY is at the session level, then MIKEY allocates e.g. the Crypto Sessions Identifiers (CS IDs, see [MIKEY)] '1' and '2' for the first m line, and '3' and '4' for the second m line.

* the possibility for MIKEY to receive incoming MIKEY messages and return a status code from/to the SIP/RTSP application.

* the possibility for the SIP or RTSP applications to receive information from MIKEY. This would typically include the receiving of the Crypto Session Bundle Identifier (CSB ID, see [MIKEY], to later be able to identify the active MIKEY session), and the SSRCs and the rollover counter (ROC, see [SRTP]) for SRTP usage. It is also RECOMMENDED that extra information about errors can be received.

* the possibility for the SIP or RTSP application to receive outgoing MIKEY messages.

* the possibility to tear down a MIKEY CSB (e.g. if the SIP session
  is closed, the CSB SHOULD also be closed).

8. Security Considerations

The framework for transfer of key management data as described here is intended to provide the security parameters for the end-to-end protection of the media session. It is furthermore good practice to secure the session setup (e.g. SDP, SIP, RTSP, SAP). However, it might be that the security of the session setup is not possible to achieve end-to-end, but only hop-by-hop. For example, SIP requires intermediate proxies to have access to part of the SIP message, and sometimes also to the SDP description (c.f. [E2M]), although end-to-end confidentiality can hide bodies from intermediaries. General security considerations for the session setup can be found in SDP [SDPnew], SIP [SIP], and RTSP [RTSP]. The framework defined in this memo is useful when the session setup is not protected in an end-to-end fashion, but the media streams needs to be end-to-end protected, hence the security parameters (such as keys) are not wanted revealed to nor manipulated by intermediaries.

The security will also depend on the level of security the key management protocol offers. It follows that, under the assumption that the key management schemes are secure, the SDP can be passed along unencrypted without affecting the key management as such, and the media streams will still be secure even if some attackers gained knowledge of the SDP contents. Further security considerations can be found for each key management protocol (for MIKEY these can be found in [MIKEY]). However, if the SDP messages are not sent integrity protected 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 may lead to undesired interruption in the setup. See also the attacks described at the end of this section.

The only integrity protected attribute of the media stream is, in the framework proposed here, the set of key management protocols. It is for instance possible to (1) swap key management offers across SDP messages, or, (2) inject a previous key management offer into a new SDP message. Making the (necessary) assumption that all involved key management protocols are secure, the second attack will be detected by replay protection mechanisms of the key management protocol(s). Making the further assumption that, according to normal best current practice, the production of each key management offer is done with independent (pseudo)random choices (for session keys and other

parameters), the first attack will either be detected in the Responder's (now incorrect) verification reply message (if such is

used), or, be a pure DoS attack, resulting in Initiator and Responder using different keys.

It is RECOMMENDED for the identity at the SPD level to be the one authenticated at the key management protocol level. This might however need to keep into consideration privacy aspects, which are out of scope for this famework.

The use of multiple key management protocols in the same offer may open up the possibility of a bidding-down attack, as specified in Section 3.1.4. To exclude such possibility, the authentication of the protocol identifier list is used. Note though, that the security level of the authenticated protocol identifier will be as high (or low), as the "weakest" protocol. Therefore, the offer MUST NOT contain any security protocols (or configurations thereof) weaker than permitted by local security policy.

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. Note also that the declined offer could be result of an attacker who sits on the path and removes all the key management offers. The bidding-down attack prevention, as described above, would not work in this case (as the answerer receives no key management attribute). Also here it is impossible to assure the authenticity of a declined offer, though here the reason is the "peeling-off" attack. It is up to the local policy to decide the behavior in the case that the response declines any security (therefore there is impossibility of authenticating it). If for example the local policy requires a secure communication and cannot accept an unsecured one, then the session setup SHALL be aborted.

9. IANA Considerations

9.1. SDP Attribute Registration

The IANA is hereby requested to create a new subregistry for the

purpose of key management protocol integration with SDP.

SDP Attribute Field ("att-field"):

Name:	key-mgmt-att-field
Long form:	key management protocol attribute field
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 RFC xxxx, Section 2.1 and 8.3.

9.2. RTSP Registration

The IANA is hereby requested to create a new subregistry for the purpose of key management protocol integration with RTSP.

Following the guidelines of [RTSP], the registration is defined as follows:

Header name:	keymgmt
Header syntax:	see RFC xxxx, Section 2.2
Intended usage:	see RFC xxxx, Section 2.2
Proxy treatment:	Proxies SHALL NOT add, change, or delete the header. The proxy does not need to read this header.
Purpose:	see RFC xxxx, Section 2

The RTSP Status-Code "463" [RFC xxxx], with the default string "Key management failure", needs to be registered.

9.3. Protocol Identifier Registration

This document defines one new name space, the "SDP/RTSP key management protocol identifier", associated with the protocol identifier, KMPID, defined in Section 2 to be used with the above registered attributes in SDP and RTSP.

The IANA is hereby requested to create a new subregistry for the KMPID parameter, with the following registration created initially: "mikey".

Value name:	mikey
Long name:	Multimedia Internet KEYing
Purpose:	Usage of MIKEY with the key-mgmt-att-field attribute and the keymgmt RTSP header
Reference:	Section 7 in RFC 3830

Note that this registration implies that the protocol identifier, KMPID, name space will be shared between SDP and RTSP.

Further values may be registered according to the "Specification Required" policy as defined in [RFC2434]. Each new registration needs to indicate the parameter name, and register it within IANA. Note that the parameter name is case sensitive and it is RECOMMENDED that the name to 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 and the requested details of interaction between the key management protocol and SDP, as specified in RFC xxxx.

New values MUST be register with IANA. Registrations SHALL include the following information:

* Contact: the contact name and email address
* Value name: the name of the value being registered (which MUST comply with the KMPID as defined in Section 2)
* Long Name: long-form name in English
* Purpose: short explanation of the purpose of the registered name.
* Reference: a reference to the specification (e.g. RFC number) providing the usage guidelines in accordance to Section 5 (and also complying to the specified requirements).

10. Acknowledgments

The authors would like to thank Francois Audet, Rolf Blom, Johan Bilien, Magnus Brolin, Erik Eliasson, Martin Euchner, Steffen Fries, Joerg Ott, Jon Peterson, and Jon-Olov Vatn. A special thanks to Colin Perkins and Magnus Westerlund, who contributed in many sections.

11. Author's Addresses

Jari Arkko
Ericsson
02420 Jorvas Phone: +358 40 5079256
Finland Email: jari.arkko@ericsson.com

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

Fredrik Lindholm
Ericsson
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

12. References

12.1. Normative References

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

[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, draft-ietf-mmusic-sdp-new-15.txt.

[SIP] Rosenberg et al., "SIP: Session Initiation Protocol", IETF, RFC 3261.

[RFC2234] Crocker, D. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[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.

12.2. Informative References

[E2M] Ono, K. and Tachimoto, S., "End-to-middle security in the Session Initiation Protocol (SIP)", Internet Draft, IETF, draft-ono-sipping-end2middle-security-03.

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

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", IETF, RFC 3851.

[SDES] Andreasen, F., Baugher, M., Wing, D., "Session Description Protocol Security Descriptions for Media Streams", work in progress, February 2005.

[SRTP] Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K., "The Secure Real-time Transport Protocol", IETF, RFC 3711.

[SPREC] Andreasen, F., Baugher, M., and Wing, D., "Security Preconditions for Session Description Protocol Media Streams", work in progress, February 2004.

Intellectual Property Statement

Copyright Notice

Disclaimer of Validity

This Internet-Draft expires in December 2005.