                    MIKEY: Multimedia Internet KEYing
                      <draft-ietf-msec-mikey-08.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

   Security protocols for real-time multimedia applications have started
   to appear. This has brought forward the need for a key management
   solution to support these protocols.

   This document describes a key management scheme that can be used for
   real-time applications (both for peer-to-peer communication and group
   communication). In particular, its use to support the Secure Real-
   time Transport Protocol is described in detail.

TABLE OF CONTENTS

1. Introduction

   There has recently been work to define a security protocol for the
   protection of real-time applications running over RTP, [SRTP].
   However, a security protocol needs a key management solution to
   exchange keys and related security parameters. There are some
   fundamental properties that such a key management scheme has to
   fulfill to serve streaming and real-time applications (such as
   unicast and multicast), in particular in heterogeneous (mix of wired
   and wireless) networks.

   This document describes a key management solution that addresses
   multimedia scenarios (e.g. SIP [SIP] calls and RTSP [RTSP] sessions).

The focus is on how to set up key management for secure multimedia
   sessions such that requirements in a heterogeneous environment are
   fulfilled.

1.1. Existing solutions

   There is work done in IETF to develop key management schemes. For
   example, IKE [IKE] is a widely accepted unicast scheme for IPsec, and
   the MSEC WG is developing other schemes, addressed to group
   communication [GDOI, GSAKMP]. For reasons discussed below, there is
   however a need for a scheme with lower latency, suitable for
   demanding cases such as real-time data over heterogeneous networks,
   and small interactive groups.

   An option in some cases might be to use [SDP], as SDP defines one
   field to transport keys, the "k=" field. However, this field cannot
   be used for more general key management purposes, as it cannot be
   extended from the current definition.

1.2. 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
   [RFC2119].

1.3. Definitions

   (Data) Security Protocol: the security protocol used to protect the
   actual data traffic. Examples of security protocols are IPsec and
   SRTP.

   Data Security Association (Data SA): information for the security
   protocol, including a TEK and a set of parameters/policies.

   Crypto Session (CS): uni- or bi-directional data stream(s), protected
   by a single instance of a security protocol. E.g. when SRTP is used,
   the Crypto Session will often contain two streams, an RTP stream and
   the corresponding RTCP which are both protected by a single SRTP
   Cryptographic Context, i.e. they share key data and the bulk of
   security parameters in the SRTP Cryptographic Context (default
   behavior in [SRTP]). In the case of IPsec, a Crypto Session would
   represent an instantiation of an IPsec SA. A Crypto Session can be
   viewed as a Data SA (as defined in [GKMARCH]) and could therefore be
   mapped to other security protocols if needed.

   Crypto Session Bundle (CSB): collection of one or more Crypto
   Sessions, which can have common TGKs (see below) and security
   parameters.

   Crypto Session ID: unique identifier for the CS within a CSB.

Crypto Session Bundle ID (CSB ID): unique identifier for the CSB.

TEK Generation Key (TGK): a bit-string agreed upon by two or more
parties, associated with CSB. From the TGK, Traffic-encrypting Keys
can then be generated without need of further communication.

Traffic-Encrypting Key (TEK): the key used by the security protocol
to protect the CS (this key may be used directly by the security
protocol or may be used to derive further keys depending on the
security protocol). The TEKs are derived from the CSB's TGK.

TGK re-keying: the process of re-negotiating/updating the TGK (and
consequently future TEK(s)).

Initiator: the Initiator of the key management protocol, not
necessarily the Initiator of the communication.

Responder: the Responder in the key management protocol.

Salting key: a random or pseudo-random (see [RAND, HAC]) string used
to protect against some off-line pre-computation attacks on the
underlying security protocol.

PRF(k,x): a keyed pseudo-random function (see [HAC]).
E(k,m):   encryption of m with the key k.
PKx: the public key of x
[]   an optional piece of information
{}   denotes zero or more occurrences
|| concatenation
| OR (selection operator)
^   exponentiation
XOR   exclusive or

Bit and byte ordering: throughout the document bits and bytes are as
usual indexed from left to right, with the leftmost bits/bytes being
the most significant.

1.4. Abbreviations

AES   Advanced Encryption Standard
CM   Counter Mode (as defined in [SRTP])
CS   Crypto Session
CSB   Crypto Session Bundle
DH   Diffie-Hellman
DoS   Denial of Service
MAC   Message Authentication Code
MIKEY   Multimedia Internet KEYing
PK   Public-Key
PSK   Pre-Shared key
RTP   Real-time Transport Protocol

```
   RTSP    Real Time Streaming Protocol
   SDP     Session Description Protocol
```

```
   SIP    Session Initiation Protocol
   SRTP   Secure RTP
   TEK    Traffic-encrypting key
   TGK    TEK Generation Key
```

1.5. Outline

Section 2 describes the basic scenarios and the design goals for
which MIKEY is intended. It also gives a brief overview of the entire
solution and its relation to the group key management architecture
[GKMARCH].

The basic key transport/exchange mechanisms are explained in detail
in Section 3. The key derivation, and other general key management
procedures are described in Section 4.

Section 5 describes the expected behavior of the involved parties.
This also includes message creation and parsing.

All definitions of the payloads in MIKEY are described in Section 6.

Section 7 deals with transport considerations, while Section 8
focuses on how MIKEY is used in group scenarios.

The Security Considerations section (Section 9), gives a deeper
explanation of important security related topics.

2. Basic Overview

2.1. Scenarios

MIKEY is mainly intended to be used for peer-to-peer, simple one-to-
many, and small-size (interactive) groups. One of the main multimedia
scenarios considered when designing MIKEY has been the conversational
multimedia scenario, where users may interact and communicate in
real-time. In these scenarios it can be expected that peers set up
multimedia sessions between each other, where a multimedia session
may consist of one or more secured multimedia streams (e.g. SRTP
streams).

```
   peer-to-peer/           many-to-many         many-to-many
    simple one-to-many         (distributed)        (centralized)
           ++++        ++++          ++++      ++++            ++++
           |. |        |A |          |B |      |A |----    ----|B |
          --| ++++     |  |----------|  |      |  |    \ /     |  |
++++     /  ++|. |     ++++          ++++      ++++    (S)     ++++
|A |---------| ++++      \           /           |
|  |    \    ++|B |       \         /            |
++++     \-----|  |        \ ++++ /            ++++
           ++++            \|C |/             |C |
                            |  |              |  |
                            ++++             ++++
```
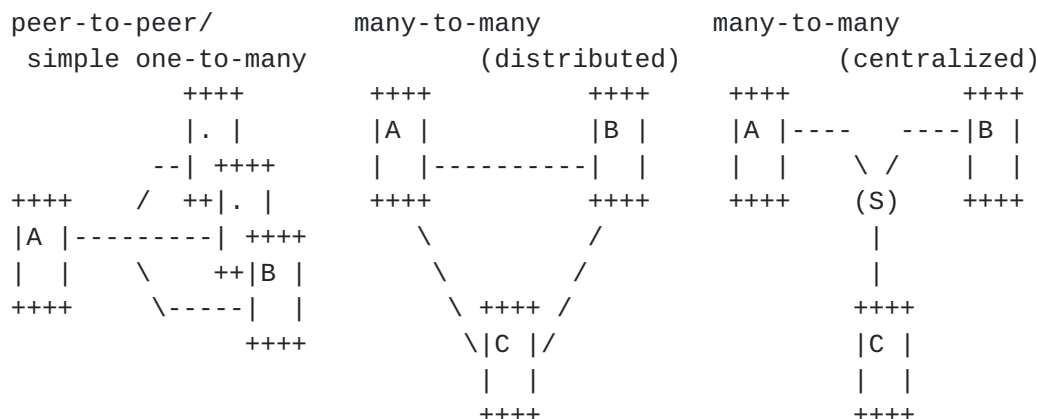
Figure 2.1: Examples of the four scenarios: peer-to-peer, simple one-
to-many, many-to-many without centralized server (also denoted as
small interactive group), and many-to-many with a centralized server.

We identify in the following some typical scenarios which involve the
multimedia applications we are dealing with (see also Figure 2.1).

a) peer-to-peer (unicast), e.g. a SIP-based [SIP] call between two
parties where it may be desirable that the security is either set up
by mutual agreement or that each party sets up the security for its
own outgoing streams.

b) simple one-to-many (multicast), e.g. real-time presentations,
where the sender is in charge of setting up the security.

c) many-to-many, without a centralized control unit, e.g. for small-
size interactive groups where each party may set up the security for
its own outgoing media. Two basic models may be used here. In the
first model, the Initiator of the group acts as the group server (and
is the only one authorized to include new members). In the second
model, authorization information to include new members can be
delegated to other participants.

d) many-to-many, with a centralized control unit, e.g. for larger
groups with some kind of Group Controller that sets up the security.

The key management solutions may be different in the above scenarios.
When designing MIKEY, the main focus has been on case a, b, and c.
For scenario c, only the first model is covered by this document.

2.2. Design Goals

The key management protocol is designed to have the following
characteristics:

* End-to-end security. Only the participants involved in the

communication have access to the generated key(s).

   * Simplicity.

   * Efficiency. Designed to have:
     - low bandwidth consumption,
     - low computational workload,
     - small code size, and
     - minimal number of roundtrips.

   * Tunneling. Possibility to "tunnel"/integrate MIKEY in session
   establishment protocols (e.g. SDP and RTSP).

   * Independent of any specific security functionality of the
   underlying transport.

2.3. System Overview

   One objective of MIKEY is to produce a Data SA for the security
   protocol, including a traffic-encrypting key (TEK), which is derived
   from a TEK Generation Key (TGK), and used as input to the security
   protocol.

   MIKEY supports the possibility to establish keys and parameters for
   more than one security protocol (or for several instances of the same
   security protocol) at the same time. The concept of Crypto Session
   Bundle (CSB) is used to denote a collection of one or more Crypto
   Sessions that can have common TGK and security parameters, but which
   obtain distinct TEKs from MIKEY.

   The procedure of setting up a CSB and creating a TEK (and Data SA),
   is done in accordance with Figure 2.2:

   1. A set of security parameters and TGK(s) are agreed upon for the
   Crypto Session Bundle (this is done by one of the three alternative
   key transport/exchange mechanisms, see Section 3).

   2. The TGK(s) is used to derive (in a cryptographically secure way) a
   TEK for each Crypto Session.

   3. The TEK, together with the security protocol parameters, represent
   the Data SA, which is used as the input to the security protocol.

```
           +-----------------+
           |       CSB       |
           |  Key transport  |                (see Section 3)
           |    /exchange    |
           +-----------------+
                 |     :
                 | TGK :
                 v     :
           +----------+ :
   CS ID ->|    TEK   | : Security protocol     (see Section 4)
           |derivation| : parameters (policies)
           +----------+ :
            TEK |     :
                 v     v
                Data SA
                  |
                  v
           +-------------------+
           |  Crypto Session   |
           |(Security Protocol)|
           +-------------------+
```
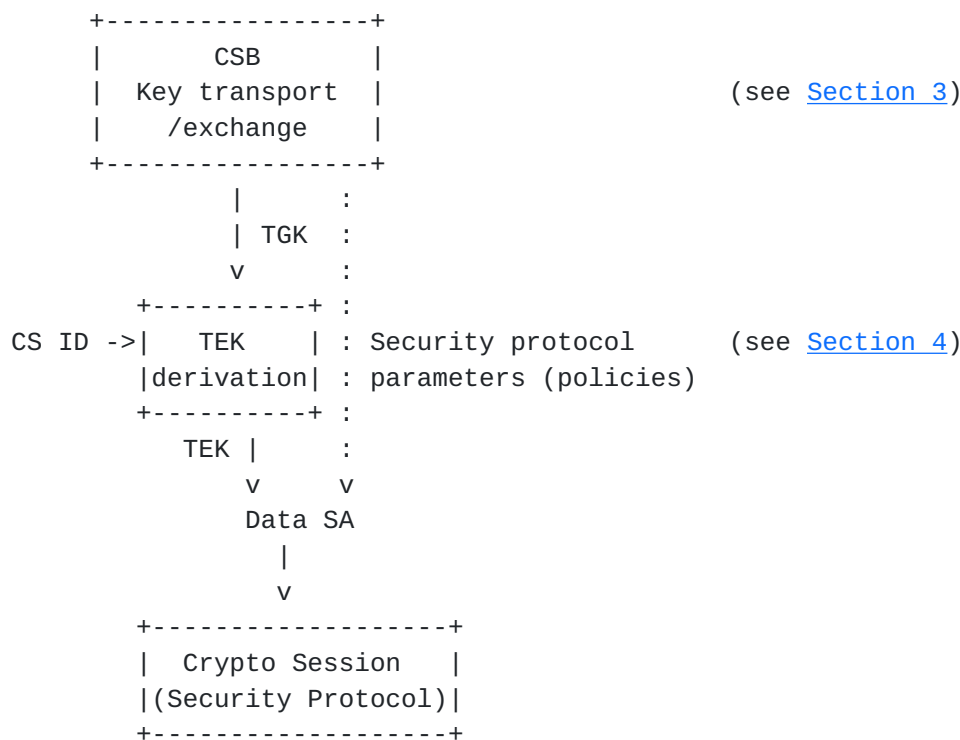
   Figure 2.2: Overview of MIKEY key management procedure.

   The security protocol can then either use the TEK directly, or, if
   supported, derive further session keys from the TEK (e.g. see SRTP
   [SRTP]). It is however up to the security protocol to define how the
   TEK is used.

   MIKEY can be used to update TEKs and the Crypto Sessions in a current
   Crypto Session Bundle (see Section 4.5). This is done by executing
   the transport/exchange phase once again to obtain a new TGK (and
   consequently derive new TEKs) or to update some other specific CS
   parameters.

2.4. Relation to GKMARCH

   The Group key management architecture (GKMARCH) [GKMARCH] describes a
   general architecture for group key management protocols. MIKEY is a
   part of this architecture, and can be used as a so-called
   Registration protocol. The main entities involved in the architecture
   are the group controller/key server (GCKS), the receiver(s), and the
   sender(s).

   In MIKEY, the sender could act as GCKS and push down keys to the
   receiver(s).

   Note that e.g., in a SIP-initiated call, the sender may also be a

receiver. As MIKEY addresses small interactive groups, a member may

dynamically change between being a sender and receiver (or being both
simultaneously).

3. Basic Key Transport and Exchange Methods

The following sub-sections define three different methods to
transport/establish a TGK: with the use of a pre-shared key, public-
key encryption, and Diffie-Hellman (DH) key exchange. In the
following we for simplicity assume unicast communication. In addition
to the TGK, a random "nonce", denoted RAND, is also transported. In
all three cases, the TGK and RAND values are then used to derive TEKs
as described in Section 4.1.3. A timestamp is also sent, to avoid
replay attacks (see Section 5.4).

The pre-shared key method and the public-key method are both based on
key transport mechanisms, where the actual TGK is pushed (securely)
to the recipient(s). In the Diffie-Hellman method, the actual TGK is
instead derived from the Diffie-Hellman values exchanged between the
peers.

The pre-shared case is, by far, the most efficient way to handle the
key transport due to the use of symmetric cryptography only. This
approach has also the advantage that only a small amount of data has
to be exchanged. Of course, the problematic issue is scalability as
it is not always feasible to share individual keys with a large group
of peers. Therefore, this case mainly addresses scenarios such as
server-to-client and also those cases where the public-key modes have
already been used thus allowing to "cache" a symmetric key (see below
and Section 3.2).

Public-key cryptography can be used to create a scalable system. A
disadvantage with this approach is that it is more resource consuming
than the pre-shared key approach. Another disadvantage is that in
most cases a PKI (Public Key Infrastructure) is needed to handle the
distribution of public keys. Of course, it is possible to use public
keys as pre-shared keys (e.g. by using self-signed certificates). It
should also be noted that, as mentioned above, this method may be
used to establish a "cached" symmetric key that later can be used to
establish subsequent TGKs by using the pre-shared key method (hence,
the subsequent request can be executed more efficiently).

The Diffie-Hellman (DH) key agreement method has in general a higher
resource consumption (both computationally and in bandwidth) than the
previous ones, and needs certificates as the public-key case.
However, it has the advantage of providing perfect forward secrecy
(PFS) and flexibility by allowing implementation in several different
finite groups.

Note that by using the DH method, the two involved parties will
generate a unique unpredictable random key. Therefore, it is not
possible to use this DH method to establish a group TEK (as the

different parties in the group would end up with different TEKs). It
is not the intention of the DH method to work in this scenario, but
to be a good alternative in the special peer-to-peer case.

The following general notation is used:

HDR: The general MIKEY header, which includes MIKEY CSB related data
(e.g. CSB ID) and information mapping to the specific security
protocol used. See Section 6.1 for payload definition.

T: The timestamp, used mainly to prevent replay attacks. See
Section 6.6 for payload definition and also Section 5.4 for other
timestamp related information.

IDx: The identity of entity x (i=Initiator, r=Responder). See
Section 6.7 for payload definition.

RAND: Random/pseudo-random byte-string, which is always included in
the first message from the Initiator. RAND is used as freshness value
for the key generation. It is not included in update messages of a
CSB. See Section 6.11 for payload definition. For randomness
recommendations for security, see [RAND].

SP: The security policies for the data security protocol. See
Section 6.10 for payload definition.

3.1. Pre-shared key

In this method, the pre-shared secret key, s, is used to derive key
material for both the encryption (encr_key) and the integrity
protection (auth_key) of the MIKEY messages, as described in Section
4.1.4. The encryption and authentication transforms are described in
Section 4.2.


```
Initiator                                  Responder

I_MESSAGE =
HDR, T, RAND, [IDi],
     {SP}, KEMAC            --->
                                       R_MESSAGE =
                            [<---]     HDR, T, [IDr], V
```


The main objective of the Initiator's message (I_MESSAGE) is to
transport one or more TGKs (carried into KEMAC) and a set of security
parameters (SPs) to the Responder in a secure manner. As the
verification message from the Responder is optional, the Initiator
indicates in the HDR whether it requires a verification message or

not from the Responder.

KEMAC = E(encr_key, {TGK}) || MAC

The KEMAC payload contains a set of encrypted sub-payloads and a MAC.
Each sub-payload includes a, by the Initiator, randomly and
independently chosen TGK (and possible other related parameters,
e.g., the key lifetime). The MAC is a Message Authentication Code
covering the entire MIKEY message using the authentication key,
auth_key. See Section 6.2 for payload definition and Section 5.2 for
exact definition of the MAC calculation.

The main objective of the verification message from the Responder is
to obtain mutual authentication. The verification message, V, is a
MAC computed over the Responder's entire message, the timestamp (the
same as the one that was included in the Initiator's message), and
the two parties identities, using the authentication key. See also
Section 5.2 for the exact definition of the Verification MAC
calculation and Section 6.9 for payload definition.

The ID fields SHOULD be included, but they MAY be left out when it
can be expected that the peer already knows the other party's ID
(otherwise it cannot look up the pre-shared key). This could e.g. be
the case if the ID is extracted from SIP.

This method is MANDATORY to implement.

3.2. Public-key encryption

```
Initiator                                        Responder

I_MESSAGE =
HDR, T, RAND, [IDi|CERTi], {SP},
    KEMAC, [CHASH], PKE, SIGNi          --->
                                             R_MESSAGE =
                                 [<---]    HDR, T, [IDr], V
```

As in the previous case, the main objective of the Initiator's
message is to transport one or more TGKs and a set of security
parameters to the Responder in a secure manner. This is done using an
envelope approach where the TGKs are encrypted (and integrity
protected) with keys derived from a randomly/pseudo-randomly chosen
"envelope key". The envelope key is sent to the Responder encrypted
with the public key of the Responder.

The PKE contains the encrypted envelope key: PKE = E(PKr, env_key).
It is encrypted using the Responder's public key (PKr). If the
Responder posses several public keys, the Initiator can indicate the
key used in the CHASH payload (see Section 6.8).

The KEMAC contains a set of encrypted sub-payloads and a MAC:

KEMAC = E(encr_key, IDi || {TGK}) || MAC

The first payload (IDi) in KEMAC is the identity of the Initiator
(not a certificate, but generally the same ID as the one specified in
the certificate). Each of the following payloads (TGK) includes a, by
the Initiator, randomly and independently chosen TGK (and possible
other related parameters, e.g., the key lifetime). The encrypted part
is then followed by a MAC, which is calculated over the KEMAC
payload. The encr_key and the auth_key are derived from the envelope
key, env_key, as specified in Section 4.1.4. See also Section 6.2 for
payload definition.

The SIGNi is a signature covering the entire MIKEY message, using the
Initiator's signature key (see also Section 5.2 for the exact
definition).

The main objective of the verification message from the Responder is
to obtain mutual authentication. As the verification message V from
the Responder is optional, the Initiator indicates in the HDR whether
it requires a verification message or not from the Responder. V is
calculated in the same way as in the pre-shared key mode (see also
Section 5.2 for the exact definition). See Section 6.9 for payload
definition.

Note that there will be one encrypted IDi and possibly also one
unencrypted IDi. The encrypted one is together with the MAC used as a
countermeasure for certain man-in-the-middle attacks, while the
unencrypted is always useful for the Responder to immediately
identify the Initiator. The encrypted IDi MUST always be verified to
be equal with the expected IDi.

It is possible to cache the envelope key, so that it can be used as a
pre-shared key. It is not recommended to cache this key indefinitely
(however it is up to the local policy to decide this). This function
may be very convenient during the lifetime of a CSB, if a new crypto
session needs to be added (or an expired one removed). Then, the pre-
shared key can be used, instead of the public keys (see also Section
4.5). If the Initiator indicates that the envelope key should be
cached, the key is at least to be cached during the lifetime of the
entire CSB.

The cleartext ID fields and certificate SHOULD be included, but they
MAY be left out when it can be expected that the peer already knows
the other party's ID, or can obtain the certificate in some other
manner. This could e.g. be the case if the ID is extracted from SIP.

For certificate handling, authorization and policies, see Section
4.3.

This method is MANDATORY to implement.

[3.3](). Diffie-Hellman key exchange

For a fixed, agreed upon, cyclic group, (G,*), we let g denote a
generator for this group. Choices for the parameters are given in
[Section 4.2.7](). The other transforms below are described in [Section
4.2]().

This method creates a DH-key, which is used as the TGK. This method
cannot be used to create group keys, only be used to create single
peer-to-peer keys. This method is OPTIONAL to implement.

```
Initiator                                      Responder

I_MESSAGE =
HDR, T, RAND, [IDi|CERTi],
     {SP}, DHi, SIGNi          --->
                                       R_MESSAGE =
                               <---    HDR, T, [IDr|CERTr], IDi,
                                       DHr, DHi, SIGNr
```

The main objective of the Initiator's message is to, in a secure way,
provide the Responder with its DH value (DHi) $g^{(xi)}$, where xi MUST
be randomly/pseudo-randomly and secretly chosen, and a set of
security protocol parameters.

The SIGNi is a signature covering the Initiator's MIKEY message,
I_MESSAGE, using the Initiator's signature key (see [Section 5.2]() for
the exact definition).

The main objective of the Responder's message is to, in a secure way,
provide the Initiator with the Responder's value (DHr) $g^{(xr)}$, where
xr MUST be randomly/pseudo-randomly and secretly chosen. The
timestamp that is included in the answer is the same as the one
included in the Initiator's message.

The SIGNr is a signature covering the Responder's MIKEY message,
R_MESSAGE, using the Responder's signature key (see [Section 5.2]() for
the exact definition).

The DH group parameters (e.g., the group G, the generator g, etc) are
chosen by the Initiator and signaled to the Responder. Both parties
calculate the TGK, $g^{(xi*xr)}$ from the exchanged DH-values.

Note that this approach does not require that the Initiator has to
posses any of the Responder's certificates before the setup. Instead,
it is sufficient that the Responder includes its signing certificate

in the response.

Arkko, et al.

The ID fields and certificate SHOULD be included, but they MAY be
left out when it can be expected that the peer already knows the
other party's ID (or can obtain the certificate in some other
manner). This could e.g. be the case if the ID is extracted from SIP.

For certificate handling, authorization and policies, see
Section 4.3.

4. Selected Key Management Functions

MIKEY manages symmetric keys in two main ways. Firstly, following key
transport or key exchange of TGK(s) (and other parameters) as defined
by any of the above three methods, MIKEY maintains a mapping between
Data SA identifiers and Data SAs, where the identifiers used depend
on the security protocol in question, see Section 4.4. Thus, when the
security protocol requests a Data SA, given such a Data SA
identifier, an up-to-date Data SA will be obtained. In particular,
correct keying material, TEK(s), might need to be derived. The
derivation of TEK(s) (and other keying material) is done from a TGK
and is described in Section 4.1.3.

Secondly, for use within MIKEY itself, two key management procedures
are needed:

* in the pre-shared case, deriving encryption and authentication key
material from a single pre-shared key, and

* in the public key case, deriving similar key material from the
transported envelope key.

These two key derivation methods are specified in section 4.1.4.

All the key derivation functionality mentioned above is based on a
pseudo-random function, defined next.

4.1. Key Calculation

We define in the following a general method (pseudo-random function)
to derive one or more keys from a "master" key. This method is used
to derive:

* TEKs from a TGK and the RAND value,

* encryption, authentication, or salting key from a pre-shared/
envelope key and the RAND value.

4.1.1. Assumptions

We assume that the following parameters are in place:

```
csb_id : Crypto Session Bundle ID (32-bits unsigned integer)
cs_id  : the Crypto Session ID (8-bits unsigned integer)
RAND   : an (at least) 128-bit (pseudo-)random bit-string sent by the
         Initiator in the initial exchange.
```

The key derivation method has the following input parameters:

```
inkey     : the input key to the derivation function
inkey_len : the length in bits of the input key
label     : a specific label, dependent on the type of the key to be
            derived, the RAND, and the session IDs
outkey_len: desired length in bits of the output key.
```

The key derivation method has the following output:

```
outkey: the output key of desired length.
```

4.1.2. Default PRF Description

Let HMAC be the SHA-1 based message authentication function, see
[HMAC], [SHA-1]. Similar to [TLS], define:

```
P (s, label, m) = HMAC (s, A_1 || label) ||
                  HMAC (s, A_2 || label) || ...
                  HMAC (s, A_m || label)
where

A_0 = label,
A_i = HMAC (s, A_(i-1))
s is the input key
m is a positive integer.
```

Values of label depend on the case in which the PRF is invoked, and
values are specified in the following for the default PRF. Thus, note
that other PRFs later added to MIKEY MAY specify different input
parameters.

The following procedure describes a pseudo-random function, denoted
PRF(inkey,label), based on the above P-function, applied to compute
the output key, outkey:

* let n = inkey_len / 512, rounded up to the nearest integer if not
  already an integer
* split the inkey into n blocks, inkey = s_1 || ... || s_n, where all
   s_i, except possibly s_n, are 512 bits each
* let m = outkey_len / 160, rounded up to the nearest integer if not
  already an integer

(The values "512" and "160" equals the input block-size and output
hash size, respectively, of the SHA-1 hash as part of the P-
function.)

Then, the output key, outkey, is obtained as the outkey_len most
significant bits of

PRF(inkey, label) = P(s_1, label, m) XOR P(s_2, label, m) XOR ...
                    XOR P(s_n, label, m).

4.1.3. Generating keys from TGK

In the following, we describe how keying material is derived from a
TGK, thus assuming that mapping of Data SA identifier to the correct
TGK has already been done according to Section 4.4.

The key derivation method SHALL be executed using the above PRF with
the following input parameters:

```
inkey       : TGK
inkey_len   : bit length of TGK
label       : constant || cs_id || csb_id || RAND
outkey_len  : bit length of the output key.
```

The constant part of label depends on the type of key that is to be
generated. The constant 0x2AD01C64 is used to generate a TEK from
TGK. If the security protocol itself does not support key derivation
for authentication and encryption from the TEK, separate
authentication and encryption keys MAY be created directly for the
security protocol by replacing 0x2AD01C64 with 0x1B5C7973 and
0x15798CEF respectively, and outkey_len by the desired key-length(s)
in each case.

A salt key can be derived from the TGK as well, by using the constant
0x39A2C14B. Note that the Key data sub-payload (Section 6.13) can
carry a salt. The security protocol in need of the salt key, SHALL
use the salt key carried in the Key data sub-payload (in the pre-
shared and public-key case), when present. If that is not sent, then
it is possible to derive the salt key via the key derivation
function, as described above.

The table below summarizes the values of constant, used to generate
keys from a TGK.

```
constant    | derived key from the TGK
--------------------------------------
0x2AD01C64  | TEK
0x1B5C7973  | authentication key
```

```
0x15798CEF  | encryption key
0x39A2C14B  | salting key
```

Table 4.1.3: Values of constant for the derivation of keys from TGK.

Note that these 32-bit constant values (listed in the table above) are taken from the decimal digits of e (i.e. 2.7182...), and where each constant consist of nine decimals digits (e.g. the first nine decimal digits 718281828 = 0x2AD01C64). The strings of nine decimal digits are not chosen at random, but as consecutive "chunks" from the decimal digits of e.

4.1.4. Generating keys for MIKEY messages from an envelope/pre-shared key

This derivation is to form the symmetric encryption key (and salting key) for the encryption of the TGK in the pre-shared key and public key methods. This is also used to derive the symmetric key used for the message authentication code in these messages, and the corresponding verification messages. Hence, this derivation is needed in order to get different keys for the encryption and the MAC (and in the case of the pre-shared key, it will result in fresh key material for each new CSB). The parameters for the default PRF are here:

```
inkey      : the envelope key or the pre-shared key
inkey_len  : the bit length of inkey
label      : constant || 0xFF || csb_id || RAND

outkey_len : desired bit length of the output key.
```

The constant part of label depends on the type of key that is to be generated from an envelope/pre-shared key, as summarized below.

```
constant    | derived key
--------------------------------------
0x150533E1  | encryption key
0x2D22AC75  | authentication key
0x29B88916  | salt key
```

Table 4.1.4: Values of constant for the derivation of keys from an envelope/pre-shared key.

4.2 Pre-defined Transforms and Timestamp Formats

This section identifies standard transforms for MIKEY. The following transforms are mandatory to implement and support in the respective case. New transforms can be added in the future (see Section 4.2.9 for further guidelines).

4.2.1 Hash functions

In MIKEY, SHA-1 is the default hash function that is MANDATORY to implement.

4.2.2 Pseudo-random number generator and PRF

A cryptographically secure random or pseudo-random number generator MUST be used for the generation of the keying material and nonces, e.g. [BMGL]. However, it is implementation specific which one to use (as the choice will not affect the interoperability).

For the key derivations, the PRF specified in Section 4.1, is MANDATORY to implement. Other PRFs MAY be added by writing standard-track RFCs specifying the PRF constructions and their exact use within MIKEY.

4.2.3 Key data transport encryption

The default and mandatory-to-implement key transport encryption is AES in counter mode, as defined in [SRTP], using a 128-bit key as derived in Section 4.1.4, and using initialization vector

IV = (S XOR (0x0000 || CSB ID || T)) || 0x0000,

where S is a 112-bit salting key, also derived as in Section 4.1.4, and where T is the 64-bit timestamp sent by the Initiator.

Note: this restricts the maximum size that can be encrypted to 2^{23} bits, which is still enough for all practical purposes [SRTP].

The NULL encryption algorithm (i.e., no encryption) can be used (but is OPTIONAL to implement). Note that this MUST NOT be used unless the underlying protocols can guarantee the security. The main reason for including this is for certain specific SIP scenarios, where SDP is protected end-to-end. For this scenario, MIKEY MAY be used with the pre-shared key method and the NULL encryption and NULL authentication algorithm (see Section 4.2.4) while relying on the security of SIP. Use this option with caution!

The AES key wrap function [AESKW] is included as an OPTIONAL to implement method. If the key wrap function is used in the public key method, the NULL MAC is RECOMMENDED as the key wrap itself will provide integrity of the encrypted content (note though that the NULL MAC SHOULD NOT be used in the pre-shared key case, as the MAC in that case covers the entire message). The 128-bit key and a 64-bit salt, S, are derived in accordance to Section 4.1.4 and the key wrap IV is then set to S.

4.2.4 MAC and Verification Message function

MIKEY uses a 160-bit authentication tag, generated by HMAC with SHA-1
as the MANDATORY to implement method, see [HMAC]. Authentication keys
are derived according to Section 4.1.4. Note that the authentication
key size SHOULD be equal to the size of the hash function's output
(e.g. for HMAC-SHA-1, a 160-bit authentication key is used) [HMAC].

The NULL authentication algorithm (i.e., no MAC) can be used together
with the NULL encryption algorithm (but is OPTIONAL to implement).
Note that this MUST NOT be used unless the underlying protocols can
guarantee the security. The main reason for including this is for
certain specific SIP scenarios, where SDP is protected end-to-end.
For this scenario, MIKEY MAY be used with the pre-shared key method
and the NULL encryption and authentication algorithm while relying on
the security of SIP. Use this option with caution!

4.2.5 Envelope Key encryption

The public key encryption algorithm applied is defined by, and
dependent on the certificate used. It is MANDATORY to support RSA
PKCS#1, v1.5, and it is RECOMMENDED to also support RSA OAEP [PSS].

4.2.6 Digital Signatures

The signature algorithm applied is defined by, and dependent on the
certificate used. It is MANDATORY to support RSA PKCS#1, v1.5, and it
is RECOMMENDED to also support RSA PSS [PSS].

4.2.7 Diffie-Hellman Groups

The Diffie-Hellman key exchange uses OAKLEY 5 [OAKLEY] as mandatory
to implement. Both OAKLEY 1 and OAKLEY 2 MAY be used (but these are
OPTIONAL to implement).

See Section 4.2.9 for the guidelines to specify a new DH Group to be
used within MIKEY.

4.2.8. Timestamps

The timestamp is as defined in NTP [NTP], i.e. a 64-bit number in
seconds relative to 0h on 1 January 1900. An implementation MUST be
aware of (and take into account) the fact that the counter will
overflow approximately every 136th year. It is RECOMMENDED that the
time is always specified in UTC.

4.2.9. Adding new parameters to MIKEY

There are two different parameter sets that can be added to MIKEY.

The first is a set of MIKEY transforms (needed for the exchange
itself), and the second is the Data SAs.

New transforms and parameters (including new policies) SHALL be added
by registering with IANA (according to [RFC2434], see also Section
10) a new number for the concerned payload, and also if necessary,
document how the new transform/parameter is used. Sometimes it might
be enough to point to an already specified document for the usage,
e.g., when adding a new already standardized hash function.

In the case of adding a new DH group, the group MUST be specified in
a companion standard-track RFC (it is RECOMMENDED that the specified
group uses the same format as used in [OAKLEY]). A number can then be
assigned by IANA for such a group to be used in MIKEY.

When adding support for a new data security protocol, the following
MUST be specified:

* A map sub-payload (see Section 6.1). This is used to be able to map
a crypto session to the right instance of the data security protocol
and possibly also to provide individual parameters for each data
security protocol.

* A policy payload, i.e., specification of parameters and supported
values.

* General guidelines of usage.

4.3. Certificates, Policies and Authorization

4.3.1. Certificate handling

Certificate handling may involve a number of additional tasks not
shown here, and effect the inclusion of certain parts of the message
(c.f. [X.509]). The following observations can, however, be made:

* The Initiator typically has to find the certificate of the
  Responder in order to send the first message. If the Initiator
  does not have the Responder's certificate already, this may
  involve one or more roundtrips to a central directory agent.

* It will be possible for the Initiator to omit its own certificate
  and rely on the Responder getting this certificate using other
  means. However, we recommend doing this, only when it is
  reasonable to expect that the Responder has cached the certificate
  from a previous connection. Otherwise accessing the certificate
  would mean additional roundtrips for the Responder as well.

* Verification of the certificates using Certificate Revocation Lists
  (CRLs) [X.509] or protocols such as OCSP [OCSP] may be necessary.

All parties in a MIKEY exchange should have a local policy which
dictates whether such checks are made, how they are made, and how

often they are made. Note that performing the checks may imply
additional messaging.

4.3.2. Authorization

In general, there are two different models for making authorization
decisions for both the Initiator and the Responder, in the context of
the applications targeted by MIKEY:

* Specific peer-peer configuration. The user has configured the
  application to trust a specific peer.

  When pre-shared secrets are used, this is pretty much the only
  available scheme. Typically, the configuration/entering of the
  pre-shared secret is taken to mean that authorization is implied.

  In some cases one could use this also with public keys, e.g. if
  two peers exchange keys offline and configure them to be used for
  the purpose of running MIKEY.

* Trusted root. The user accepts all peers that can prove to have a
  certificate issued by a specific CA. The granularity of
  authorization decisions is not very precise in this method.

  In order to make this method possible, all participants in the
  MIKEY protocol need to configure one or more trusted roots. The
  participants also need to be capable of performing certificate
  chain validation, and possibly transfer more than a single
  certificate in the MIKEY messages (see also Section 6.7).

In practice, a combination of both mentioned methods might be
advantageous. Also, the possibility for a user to explicitly exclude
a specific peer (or sub tree) in a trust chain might be needed.

These authorization policies address the MIKEY scenarios a-c of
Section 2.1, where the Initiator acts as the group owner and who is
also the only one that can invite others. This implies that for each
Responder, the distributed keys MUST NOT be re-distributed to other
parties.

In a many-to-many situation, where the group control functions are
distributed (and/or where it is possible to delegate the group
control function to others), there MUST exist means to distribute
authorization information about who may be added to the group.
However, it is out of scope for this document to specify how this
should be done.

For any broader communication situation, an external authorization
infrastructure may be used (following the assumptions of [GKMARCH]).

4.3.3. Data Policies

Included in the message exchange, policies (i.e., security
parameters) for the Data security protocol are transmitted. The
policies are defined in a separate payload and are specific to the
security protocol (see also Section 6.10). Together with the keys,
the validity period of these can also be specified. This can be done
e.g., with an SPI (or SRTP MKI) or with an Interval (e.g. a sequence
number interval for SRTP), depending on the security protocol.

New parameters can be added to a policy by documenting how they
should be interpreted by MIKEY and also by registering new values in
the appropriate name space in IANA. If a completely new policy is
needed, see Section 4.2.9 for guidelines.

4.4. Retrieving the Data SA

The retrieval of a Data SA will depend on the security protocol, as
different security protocols will have different characteristics.
When adding support for a security protocol to MIKEY, some interface
of how the security protocol retrieves the Data SA from MIKEY MUST be
specified (together with policies that can be negotiated etc.).

For SRTP the SSRC (see [SRTP]) is one of the parameters used to
retrieve the Data SA (and e.g. the MKI may be used to indicate the
TGK/TEK used for the Data SA). However, the SSRC is not sufficient.
For the retrieval of the Data SA from MIKEY, it is RECOMMENDED that
the MIKEY implementation support a lookup using destination network
address and port together with SSRC. Note that MIKEY does not send
network addresses or ports. One reason for this is that they may not
be known in advance, as well as if a NAT exists in-between, problems
may arise. When SIP or RTSP is used, the local view of the
destination address and port can be obtained from either SIP or RTSP.
MIKEY can then use these addresses as the index for the Data SA
lookup.

4.5. TGK re-keying and CSB updating

MIKEY provides the means to update the CSB (e.g. transporting a new
TGK/TEK or adding a new Crypto Session to the CSB). The updating of
the CSB is done by executing MIKEY again e.g. before a TEK expires,
or when a new Crypto Session is added to the CSB. Note that MIKEY
does not provide re-keying in the GKMARCH sense, only updating of the
keys by normal unicast messages.

When MIKEY is executed again to update the CSB, it is not necessary
to include certificates and other information that was provided in
the first exchange, i.e. all payloads that are static or optional to
include may be left out (see Figure 4.1).

The new message exchange MUST use the same CSB ID as the initial
exchange, but MUST use a new timestamp. A new RAND MUST NOT be
included in the message exchange (the RAND will only have effect in
the Initial exchange). New Crypto Sessions are added if desired in
the update message. Note that a MIKEY update message does not need to
contain new keying material (i.e., new TGK). In this case the crypto
session continues to use the previously established keying material,
while updating the new information.

As explained in Section 3.2, the envelope key can be "cached" as a
pre-shared key (this is indicated by the Initiator in the first
message sent). If so, the update message is a pre-shared key message
(with the cached envelope key as the pre-shared key), i.e., it MUST
NOT be a public key message. If the public key message is used, but
the envelope key is not cached, the Initiator MUST provide a new
encrypted envelope key that can be used in the verification message.
However, the Initiator does not need to provide any other keys.

Figure 4.1 visualizes the update messages that can be sent, including
the optional parts. The big difference from the original message is
mainly that it is optional to include TGKs (or DH values in the DH
method). See also Section 3 for more details of the specific methods.

By definition, a CSB can contain several CSs. A problem that then
might occur is to synchronize the TGK re-keying if an SPI (or similar
functionality, e.g., MKI in [SRTP]) is not used. It is therefore
RECOMMENDED that an SPI or MKI is used, if more than one CS is used.

    Initiator   Responder

    Pre-shared key method:

    I_MESSAGE =
    HDR, T, [IDi], {SP}, KEMAC   --->
                                                  R_MESSAGE =
                                         [<---]   HDR, T, [IDr], V

    Public key method:

    I_MESSAGE =
    HDR, T, [IDi|CERTi], {SP}, [KEMAC],
         [CHASH], PKE, SIGNi   --->
                                                  R_MESSAGE =
                                         [<---]   HDR, T, [IDr], V

Arkko, et al.   [Page 24]

       DH method:

       I_MESSAGE =
       HDR, T, [IDi|CERTi], {SP},
            [DHi], SIGNi   --->
                                                      R_MESSAGE =
                                              <--- HDR, T, [IDr|CERTr], IDi,
                                                         [DHr, DHi], SIGNr

       Figure 4.1: Update messages.

       Note that for the DH method, if the Initiator includes the DHi
       payload, then the Responder MUST include DHr and DHi. If the
       Initiator does not include DHi, the Responder MUST NOT include DHr,
       DHi.

[5](). Behavior and message handling

       Each message that is sent by the Initiator or the Responder is built
       by a set of payloads. This section describes how messages are created
       and also when they can be used.

[5.1](). General

[5.1.1](). Capability Discovery

       The Initiator indicates the security policy to use (i.e. in terms of
       security protocol algorithms etc). If the Responder does not support
       it (for some reason), the Responder can together with an error
       message (indicating that it does not support the parameters), send
       back its own capabilities (negotiation) to let the Initiator choose a
       common set of parameters. This is done by including one or more
       security policy payloads in the error message sent in answer (see
       [Section 5.1.2]().). Multiple attributes can be provided in sequence in
       the response. This is done to reduce the number of roundtrips as much
       as possible (i.e. in most cases, where the policy is accepted the
       first time, one roundtrip is enough). If the Responder does not
       accept the offer, the Initiator must go out with a new MIKEY message.

       If the Responder is not willing/capable to provide security or the
       parties simply cannot agree, it is up to the parties' policies how to
       behave, i.e. accept an insecure communication or reject it.

       Note that it is not the intention of this protocol to have a very
       broad variety of options, as it is assumed that it should not be too
       common that an offer is denied.

       In the one-to-many and many-to-many scenarios using multicast

communication, one issue is of course that there MUST be a common

security policy to all the receivers. This limits the possibility for negotiation.

5.1.2. Error Handling

All errors due to the key management protocol SHOULD be reported to the peer(s) by an error message. The Initiator SHOULD therefore always be prepared to receive such message from the Responder.

If the Responder does not support the set of parameters suggested by the Initiator, the error message SHOULD include the supported parameters (see also Section 5.1.1).

The error message is formed as:

HDR, T, {ERR}, {SP}, [V|SIGNr]

Note that if the failure is due to the inability to authenticate the peer, the error message is OPTIONAL, and does not need to be authenticated. It is up to the local policy how to treat this kind of messages. However, if a signed error message in response to a failed authentication is returned this can be used for DoS purposes (against the Responder). Similarly, an unauthenticated error message could be sent to the Initiator in order to fool her to tear down the CSB. It is highly RECOMMENDED that the local policy takes this into consideration. Therefore, in case of authentication failure, one advice would be not to authenticate such an error message, and when receiving an unauthenticated error message only see it as a recommendation of what may have gone wrong.

5.2. Creating a message

To create a MIKEY message, a Common Header payload is first created. This payload is then followed, depending on the message type, by a set of information payloads (e.g. DH-value payload, Signature payload, Security Policy payload). The defined payloads and the exact encoding of each payload are described in Section 6.

```
                         1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  version     !  data type   ! next payload !               !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...             +
   ~                    Common Header...                         ~
   !                                                             !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! next payload  !   Payload 1 ...                             !
   +-+-+-+-+-+-+-+-+                                             +
   ~                                                             ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   :                             :                             :
   :                             :                             :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! next payload  !   Payload x ...                             !
   +-+-+-+-+-+-+-+-+                                             +
   ~                                                             ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                    MAC/Signature                            ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.1. MIKEY payload message example. Note that the payloads are byte aligned and not 32-bit aligned.

The process of generating a MIKEY message consists of the following steps:

* Create an initial MIKEY message starting with the Common Header payload.

* Concatenate necessary payloads to the MIKEY message (see the exchange definitions for payloads that may be included, and recommended order).

* As a last step (for messages that must be authenticated, this also include the verification message), create and concatenate the MAC/signature payload without the MAC/signature field filled in (if a Next payload field is included in this payload, it is set to Last payload).

* Calculate the MAC/signature over the entire MIKEY message, except the MAC/Signature field, and add the MAC/signature in the field. In the case of the verification message, the Identity_i || Identity_r || Timestamp MUST follow directly after the MIKEY message in the Verification MAC calculation. Note that the identities and the timestamp that are added are identical to those transported in the ID

and T payloads.

In the public key case, the Key data transport payload is generated by concatenating the IDi with the TGKs. This is then encrypted and placed in the data field. The MAC is calculated over the entire Key data transport payload except the MAC field. Before calculating the MAC, the Next payload field is set to zero.

Note that all messages from the Initiator MUST use a unique timestamp. The Responder does not create a new timestamp, but uses the timestamp used by the Initiator.

5.3. Parsing a message

In general, parsing of a MIKEY message is done by extracting payload by payload and checking that no errors occur. The exact procedure is implementation specific; however, for the Responder, it is RECOMMENDED that the following procedure is followed:

* Extract the Timestamp and check that it is within the allowable clock skew (if not, discard the message). Also check the replay cache (Section 5.4) so that the message is not replayed (see also Section 5.4). If the message is replayed, discard it.

* Extract ID and authentication algorithm (if not included, assume the default one).

* Verify the MAC/signature.

* If the authentication is not successful, an Auth failure Error message MAY be sent to the Initiator. The message is then discarded from further processing. See also Section 5.1.2 for treatment of errors.

* If the authentication is successful, the message is processed and also added to the replay cache. How it is processed is implementation specific. Note also that it is only successfully authenticated messages that are stored in the replay cache.

* If any unsupported parameters or errors occur during the processing, these MAY be reported to the Initiator by sending an error message. The processing is then aborted. The error message can also include payloads to describe the supported parameters.

* If the processing was successful and in case the Initiator requested it, a verification/ response message MAY be created and sent to the Initiator.

5.4. Replay handling and timestamp usage

MIKEY does not use a challenge-response mechanism for replay

handling; instead timestamps are used. This requires that the clocks
are synchronized. The required synchronization is dependent on the

number of messages that can be cached (note though, that the replay
cache only contain messages that have been successfully
authenticated). If we could assume an unlimited cache, the terminals
would not need to be synchronized at all (as the cache could then
contain all previous messages). However, if there are restrictions on
the size of the replay cache, the clocks will need to be synchronized
to some extent. In short, one can in general say that it is a
tradeoff between the size of the replay cache and the required
synchronization.

Timestamp usage prevents against replay attacks under the following
assumptions:

* Each host has a clock which is at least "loosely synchronized" to
the clocks of the other hosts.

* If the clocks are to be synchronized over the network, a secure
network clock synchronization protocol SHOULD be used, e.g. [ISO3].

* Each Responder utilizes a replay cache in order to remember the
successfully authenticated messages presented within an allowable
clock skew (which is set by the local policy).

* Replayed and outdated messages, i.e., messages that can be found in
the replay cache or which have an outdated timestamp, are discarded
and not processed.

* If the host loses track of the incoming requests (e.g. due to
overload), it rejects all incoming requests until the clock skew
interval has passed.

In a client-server scenario, servers may encounter high workload,
especially if a replay cache is needed. However, servers that assume
the role of Initiators of MIKEY will not need to manage any
significant replay cache as they will refuse all incoming messages
that are not a response to a message previously sent by the server.

In general, a client may not expect a very high load of incoming
messages and may therefore allow the degree of looseness to be on the
order of several minutes to hours. If a (D)DoS attack is launched and
the replay cache grows too large, MIKEY MAY dynamically decrease the
looseness so that the replay cache becomes manageable. However, note
that such (D)DoS can only be performed by peers that can authenticate
themselves (hence, such attack is very easy to trace and mitigate).

The maximum number of messages that a client will need to cache may
vary depending on the capacity of the client itself and the network,
but also the number of expected messages should be taken into
account.

For example, assume that we can at most spend 6kB on a replay cache.
Assume further that we need to store 30 bytes for each incoming
authenticated message (the hash of the message is 20 bytes). This
implies that it is possible to cache approximately 204 messages. If
the expected number of messages per minute can be estimated, the
clock skew can easily be calculated. E.g., in a SIP scenario where
the client is expected in the most extreme case to receive 10 calls
per minute, the clock skew needed is then approximately 20 minutes.
In a not so extreme setting, where one could expect an incoming call
every 5th minute, this would result in a clock skew on the order of
16.5 hours (approx 1000 minutes).

Consider a very extreme case, where the maximum number of incoming
messages are assumed to be on the order of 120 messages per minute,
and a requirement that the clock skew is on the order of 10 minutes,
a 48kB replay cache would be required.

Hence, one can note that the required clock skew will depend very
much on the setting in which MIKEY is used. One recommendation is to
fix a size for the replay cache, and let the allowable clock skew be
large (the initial clock skew can be set depending on the application
in which it is used). As the replay cache grows, the clock skew is
decreased depending on how many percent of the replay cache that are
used. Note that this is locally handled, which will not require
interaction with the peer (even though it may indirectly affect the
peer). Exactly how to implement such functionality is however out of
the scope of this document and considered implementation specific.

In case of a DoS attack, the client will most likely be able to
handle the replay cache. A more likely (and serious) DoS attack is a
CPU DoS attack where the attacker sends messages to the peer, which
then needs to engage resources on verifying MACs/signatures of the
incoming messages.

6. Payload Encoding

This section describes in detail all the payloads. For all encoding,
network byte order is always used. While defining supported types,
for example which hash functions are supported, the mandatory-to-
implement are indicated (as Mandatory), as well as the default (note,
default also implies mandatory to implement). The other types are
implicitly assumed optional to support.

Note that in the following the support for SRTP [SRTP] as security
protocol is defined. This will help better understanding the purpose
of the different payloads and fields. Other security protocol MAY be
specified to use within MIKEY, see Section 10.

In the following, the sign ~ indicates variable length field.

6.1. Common Header payload (HDR)

   The Common Header payload MUST always be present as the first payload
   in each message. The Common Header includes general description of
   the exchange message.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  version     !  data type    ! next payload  !V! PRF func    !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                           CSB ID                             !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! #CS          ! CS ID map type! CS ID map info                ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * version (8 bits): the version number of MIKEY.

     version = 0x01 refers to MIKEY as defined in this document.

   * data type (8 bits): describes the type of message (e.g. public-key
   transport message, verification message, error message).

```
   Data type      | Value | Comment
   ---------------------------------------
   Pre-shared     |     0 | Initiator's pre-shared key message
   PSK ver msg    |     1 | Verification message of a Pre-shared
                  |       | key message
   Public key     |     2 | Initiator's public-key transport message
   PK ver msg     |     3 | Verification message of a public-key
                  |       | message
   D-H init       |     4 | Initiator's DH exchange message
   D-H resp       |     5 | Responder's DH exchange message
   Error          |     6 | Error message
```

     Table 6.1.a

   * next payload (8 bits): identifies the payload that is added after
   this payload.

```
   Next payload  | Value | Section
   -----------------------------
   Last payload  |     0 | -
   KEMAC         |     1 | 6.2
   PKE           |     2 | 6.3
```

```
DH              |     3 | 6.4
SIGN            |     4 | 6.5
```

```
   T              |      5 | 6.6
   ID             |      6 | 6.7
   CERT           |      7 | 6.7
   CHASH          |      8 | 6.8
   V              |      9 | 6.9
   SP             |     10 | 6.10
   RAND           |     11 | 6.11
   ERR            |     12 | 6.12
   Key data       |     20 | 6.13
   General Ext.   |     21 | 6.15
```

   Table 6.1.b

   Note that some of the payloads cannot come right after the header
   (such as "Last payload", "Signature", etc.). However, the Next
   payload field is generic for all payloads. Therefore, a value is
   allocated for each payload. The Next payload field is set to zero
   (Last payload) if the current payload is the last payload.

   * V (1 bit): flag to indicate whether a verification message is
   expected or not (this has only meaning when it is set by the
   Initiator). The V flag SHALL be ignored by the receiver in the DH
   method (as the response is MANDATORY).

```
     V = 0  ==> no response expected
     V = 1  ==> response expected
```


   * PRF func (7 bits): indicates the PRF function that has been/will be
   used for key derivation.

```
     PRF func       | Value | Comments
     -------------------------------------------------------------
     MIKEY-1        |     0 | Mandatory (see Section 4.1.3)
```

   Table 6.1.c

   * CSB ID (32 bits): identifies the CSB. It is RECOMMENDED that it is
   chosen at random by the Initiator. This ID MUST be unique between
   each Initiator-Responder pair, i.e., not globally unique. An
   Initiator MUST check for collisions when choosing the ID (if the
   Initiator already has one or more established CSB with the
   Responder). The Responder uses the same CSB ID in the response.

   * #CS (8 bits): indicates the number of Crypto Sessions that will be

handled within the CBS. Note that even though it is possible to use
255 CSs, it is not likely that a CSB will include this many CSs. The

   integer 0 is interpreted as no CS included. This may be the case in
   an initial setup message.

   * CS ID map type (8 bits): specifies the method to uniquely map
   Crypto Sessions to the security protocol sessions.

     CS ID map type | Value

     SRTP-ID | 0

     Table 6.1.d

   * CS ID map info (16 bits): identifies the crypto session(s) that the
   SA should be created for. The currently defined map type is the SRTP-
   ID (defined in Section 6.1.1).

6.1.1. SRTP ID

```
                         1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! Policy_no_1   ! SSRC_1                                        !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! SSRC_1 (cont) ! ROC_1                                         !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! ROC_1 (cont)  ! Policy_no_2    ! SSRC_2                       !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! SSRC_2 (cont)                  ! ROC_2                        !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! ROC_2 (cont)                   !                             :
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ...
    :                               :                             :
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! Policy_no_#CS !            SSRC_#CS                           !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    !SSRC_#CS (cont)!            ROC_#CS                            !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! ROC_#CS (cont)!
    +-+-+-+-+-+-+-+-+
```

   * Policy_no_i (8 bits): The security policy applied for the stream
   with SSRC_i. The same security policy may apply for all CSs.

   * SSRC_i (32 bits): specifies the SSRC that MUST be used for the i-th
   SRTP stream. Note that it is the sender of the streams who chooses
   the SSRC. Therefore, it might be that the Initiator of MIKEY can not
   fill in all fields. In this case, SSRCs that are not chosen by the

Initiator are set to zero and the Responder fills in these fields in
the response message. Note that SRTP specifies requirements on the

uniqueness of the SSRCs (to avoid two-time pad problems if the same
TEK is used for more than one stream), see [SRTP].

* ROC_i (32 bits): Current rollover counter used in SRTP. If the SRTP
session has not started, this field is set to 0. This field is used
to be able for a member to join and synchronize to an already started
stream.

NOTE: The stream using SSRC_i will also have Crypto Session ID equal
to no i (NOT to the SSRC).

6.2. Key data transport payload (KEMAC)

The Key data transport payload contains encrypted Key data sub-
payloads (see Section 6.13 for definition of the Key data sub-
payload). It may contain one or more Key data payloads each including
e.g. a TGK. The last Key data payload has its Next payload field set
to Last payload. For an update message (see also Section 4.5), it is
allowed to skip the Key data sub-payloads (which will result in that
the Encr data len is equal to 0).

Note that the MAC coverage depends on the method used, i.e. pre-
shared vs public key, see below.

If the transport method used is the pre-shared key method, this Key
data transport payload is the last payload in the message (note that
the Next payload field is set to Last payload). The MAC is then
calculated over the entire MIKEY message following the directives in
Section 5.2.

If the transport method used is the public-key method, the
Initiator's identity is added in the encrypted data. This is done by
adding the ID payload as the first payload, which then is followed by
the Key data sub-payloads. Note that for an update message, the ID is
still sent encrypted to the Responder (this is to avoid certain re-
direction attacks) even though no Key data sub-payload is added
after.

The coverage of the MAC field is in the public-key case over the Key
data transport payload only, instead of the complete MIKEY message,
as in the pre-shared case. The MAC is therefore calculated over the
Key data transport payload except the MAC field and where the Next
payload field has been set to zero (see also Section 5.2).

```
                             1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ! Next payload  ! Encr alg      ! Encr data len                !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      !                          Encr data                           ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ! Mac alg       !       MAC                                     ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * Next payload (8 bits): identifies the payload that is added after
   this payload. See Section 6.1 for defined values.

   * Encr alg (8 bits): the encryption algorithm used to encrypt the
   Encr data field.

```
      Encr alg       | Value | Comment
      --------------------------------------------
      NULL           |     0 | Very restricted usage, see Section 4.2.3!
      AES-CM-128     |     1 | Mandatory ; AES-CM using a 128-bit key, see
                             |    Section 4.2.3)
      AES-KW-128     |     2 | AES Key Wrap using a 128-bit key, see
                             |    Section 4.2.3
```

      Table 6.2.a

   * Encr data len (16 bits): length of Encr data (in bytes).

   * Encr data (variable length): the encrypted key sub-payloads (see
   Section 6.13).

   * MAC alg (8 bits): specifies the authentication algorithm used.

```
      MAC alg        | Value | Comments                   | Length (bits)
      ------------------------------------------------------------------
      NULL           |     0 | restricted usage (Sec 4.2.4)| 0
      HMAC-SHA-1-160 |     1 | Mandatory, Section 4.2.4    | 160
```

      Table 6.2.b

   * MAC (variable length): the message authentication code of the
   entire message.

6.3. Envelope data payload (PKE)

   The Envelope data payload contains the encrypted envelope key that is

used in the public-key transport to protect the data in the Key data

transport payload. The encryption algorithm used is implicit from the certificate/public key used.
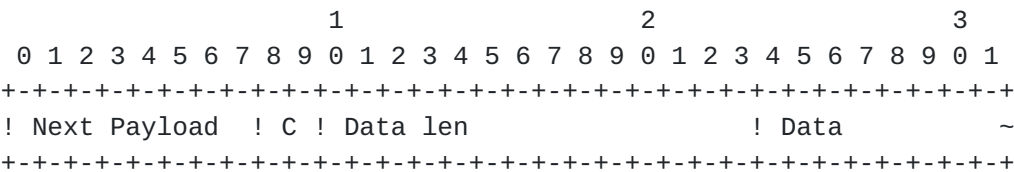
```
                       1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  ! C ! Data len                  ! Data        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after this payload. See Section 6.1 for values.

* C (2 bits): envelope key cache indicator (Section 3.2).

```
   Cache type    | Value | Comments
   --------------------------------------
   No cache      |     0 | The envelope key MUST NOT be cached
   Cache         |     1 | The envelope key MUST be cached
   Cache for CSB |     2 | The envelope key MUST be cached, but only
                 |       | to be used for the specific CSB.
   Table 6.3
```

* Data len (14 bits): the length of the data field (in bytes).

* Data (variable length): the encrypted envelope key.

6.4. DH data payload (DH)

The DH data payload carries the DH-value and indicates the DH-group used. Notice that in this sub-section "MANDATORY" is conditioned upon DH being supported at all.
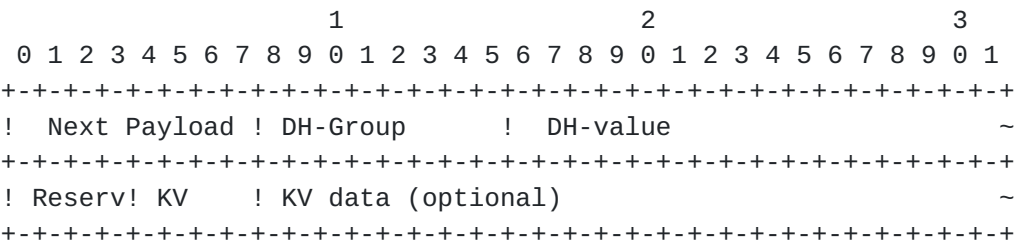
```
                       1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   Next Payload ! DH-Group     !  DH-value                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Reserv! KV    ! KV data (optional)                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after this payload. See Section 6.1 for values.

* DH-Group (8 bits): identifies the DH group used.

```
   DH-Group      | Value | Comment       | DH Value length (bits)
   -------------------------------------|---------------------
   OAKLEY 5      |     0 | Mandatory     | 1536
```

```
OAKLEY 1          |       1 |                     |    768
OAKLEY 2          |       2 |                     |   1024
```

Table 6.4

* DH-value (variable length): the public DH-value (the length is
implicit from the group used).

* KV (4 bits): indicates the type of key validity period specified.
This may be done by using an SPI (alternatively an MKI) or by
providing an interval in which the key is valid (e.g. in the latter
case, for SRTP this will be the index range where the key is valid).
See Section 6.13 for pre-defined values.

* KV data (variable length): This includes either the SPI/MKI or an
interval (see Section 6.14). If KV is NULL, this field is not
included.

6.5. Signature payload (SIGN)

The Signature payload carries the signature and its related data. The
signature payload is always the last payload in the PK transport and
DH exchange messages. The signature algorithm used is implicit from
the certificate/public key used.

```
                      1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! S type| Signature len        ! Signature                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* S type (4 bits): indicates the signature algorithm applied by
signer.

```
   S type        | Value | Comments
   -------------------------------------
   RSA/PKCS#1/1.5|    0 | Mandatory, PKCS #1 version 1.5 signature
                        [PSS]
   RSA/PSS       |    1 | RSASSA-PSS signature [PSS]
```

Table 6.5

* Signature len (12 bits): the length of the signature field (in
bytes).

* Signature (variable length): the signature (its formatting and
padding depend on the type of signature).

6.6. Timestamp payload (T)

The timestamp payload carries the timestamp information.

```
                          1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Next Payload  !   TS type     ! TS value                     ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * Next payload (8 bits): identifies the payload that is added after
   this payload. See Section 6.1 for values.

   * TS type (8 bits): specifies the timestamp type used.

TS type	Value	Comments	length of TS value
NTP-UTC	0	Mandatory	64-bits
NTP	1	Mandatory	64-bits
COUNTER	2	Optional	32-bits

   Table 6.6

   Note: COUNTER SHALL be padded (with leading zeros) to 64-bit value
   when used as input to the default PRF.

   * TS-value (variable length): The timestamp value of the specified TS
   type.

6.7. ID payload (ID) / Certificate payload (CERT)

   Note that the ID payload and the Certificate payload are two
   completely different payloads (having different payload identifiers).
   However, as they share the same payload structure they are described
   in the same section.

   The ID payload carries a uniquely defined identifier.

   The certificate payload contains an indicator of the certificate
   provided as well as the certificate data. If a certificate chain is
   to be provided, each certificate in the chain should be included in a
   separate CERT payload.

```
                          1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  Next Payload ! ID/Cert Type  ! ID/Cert len                  !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                      ID/Certificate Data                     ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
  this payload. See Section 6.1 for values.

If the payload is an ID payload the following values applies for the
ID type field:

* ID Type (8 bits): specifies the identifier type used.

   ID Type | Value | Comments

   NAI | 0 | Mandatory (see [NAI])
   URI | 1 | Mandatory (see [URI])

   Table 6.7.a

If the payload is an Certificate payload the following values applies
for the Cert type field:

* Cert Type (8 bits): specifies the certificate type used.

   Cert Type | Value | Comments

   X.509v3 | 0 | Mandatory
   X.509v3 URL | 1 | plain ASCII URL to the location of the Cert
   X.509v3 Sign | 2 | Mandatory (used for signatures only)
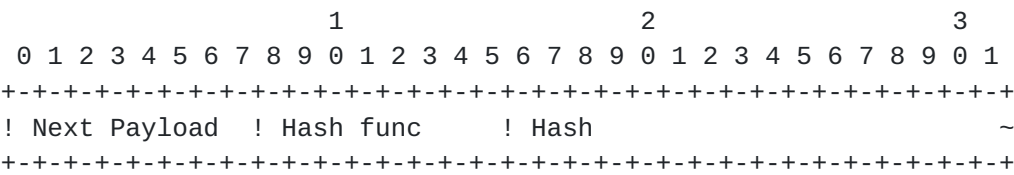   X.509v3 Encr | 3 | Mandatory (used for encryption only)

   Table 6.7.b

* ID/Cert len (16 bits): the length of the ID or Certificate field
(in bytes).

* ID/Certificate (variable length): The ID or Certificate data. The
X.509 [X.509] certificates are included as a bytes string using DER
encoding as specified in X.509.

6.8. Cert hash payload (CHASH)

The Cert hash payload contains the hash of the certificate used.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  ! Hash func     ! Hash                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload. See Section 6.1 for values.

* Hash func (8 bits): indicates the hash function that is used (see
  also Section 4.2.1).

```
Hash func      | Value | Comment      | hash length (bits)
------------------------------------------------------
SHA-1          |     0 | Mandatory    |  160
MD5            |     1 |              |  128
```

Table 6.8

* Hash (variable length): the hash data. The hash length is implicit
from the hash function used.

6.9. Ver msg payload (V)

The Ver msg payload contains the calculated verification message in
the pre-shared key and the public-key transport methods. Note that
the MAC is calculated over the entire MIKEY message as well as the
IDs and Timestamp (see also Section 5.2).

```
                        1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  ! Auth alg      ! Ver data                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload. See Section 6.1 for values.

* Auth alg (8 bits): specifies the MAC algorithm used for the
verification message. See Section 6.2 for defined values.

* Ver data (variable length): the verification message data. The
length is implicit from the authentication algorithm used.

6.10. Security Policy payload (SP)

The Security Policy payload defines a set of policies that applies to
a specific security protocol.

```
                        1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next payload  ! Policy no     ! Prot type     ! Policy param  ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~ length (cont) ! Policy param                                 ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload. See Section 6.1 for values.

   * Policy no (8 bits): each security policy payload must be given a
   distinct number for the current MIKEY session by the local peer. This
   number is used to be able to map a crypto session to a specific
   policy (see also Section 6.1.1).

   * Prot type (8 bits): defines the security protocol.

     Prot type | Value |

     SRTP | 0 |

     Table 6.10

   * Policy param length (16 bits): defines the total length of the
   policy parameters for the specific security protocol.

   * Policy param (variable length): defines the policy for the specific
   security protocol.

   The Policy param part is built up by a set of Type/Length/Value
   fields. For each security protocol, a set of possible types/values
   that can be negotiated is defined.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Type          ! Length        ! Value                        ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * Type (8 bits): specifies the type of the parameter.

   * Length (8 bits): specifies the length of the Value field (in
   bytes).

   * Value (variable length): specifies the value of the parameter.

6.10.1. SRTP policy

   This policy specifies the parameters for SRTP and SRTCP. The
   types/values that can be negotiated are defined by the following
   table:

     Type | Meaning | Possible values
     ----
        0 | Encryption algorithm | see below
        1 | Session Encr. key length | depends on cipher used
        2 | Authentication algorithm | see below

```
      3 | Session Auth. key length    | depends on MAC used
      4 | Session Salt key length     | see [SRTP] for recommendations
```

```
    5 | SRTP Pseudo Random Function | see below
    6 | Key derivation rate         | see [SRTP] for recommendations
    7 | SRTP encryption off/on       | 0 if off, 1 if on
    8 | SRTCP encryption off/on      | 0 if off, 1 if on
    9 | sender's FEC order           | see below
   10 | SRTP authentication off/on   | 0 if off, 1 if on
   11 | Authentication tag length    | in bytes
   12 | SRTP prefix length           | in bytes
```

   Table 6.10.1.a

   Note that if a Type/Value is not set, the default one is used
   (according to SRTPs own criteria).

   For the Encryption algorithm, it is enough with a one byte length and
   the currently defined possible Values are:

```
   SRTP encr alg | Value
   ---------------------
   NULL          |    0
   AES-CM        |    1
   AES-F8        |    2
```

   Table 6.10.1.b

   where AES-CM is AES in CM, and AES-F8 is AES in f8 mode [SRTP].

   For the Authentication algorithm, it is enough with a one byte length
   and the currently define possible Values are:

```
   SRTP auth alg | Value
   ---------------------
   NULL          |    0
   HMAC-SHA-1    |    1
```

   Table 6.10.1.c

   For the SRTP pseudo-random function, it is also enough with a one
   byte length and the currently define possible Values are:

```
   SRTP PRF      | Value
   ---------------------
   AES-CM        |    0
```

   Table 6.10.1.d

If FEC is used at the same time as SRTP is used, MIKEY can negotiate
the order in which these should be applied at the sender side.

```
FEC order      | Value | Comments
-------------------------------
FEC-SRTP       |     0 | First FEC, then SRTP
```

Table 6.10.1.e

6.11. RAND payload (RAND)

The RAND payload consists of a (pseudo-)random bit-string. The RAND
MUST be independently generated per CSB (note that the if a CSB has
several members, the Initiator MUST use the same RAND to all the
members). For randomness recommendations for security, see [RAND].

```
                      1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next payload  ! RAND len      ! RAND                         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload. See Section 6.1 for values.

* RAND len (8 bits): length of the RAND (in bytes). It SHOULD be at
least 16.

* RAND (variable length): a (pseudo-)randomly chosen bit-string.

6.12. Error payload (ERR)

The Error payload is used to specify the error(s) that may have
occurred.

```
                      1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Next Payload ! Error no      !           Reserved           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload. See Section 6.1 for values.

* Error no (8 bits): indicates the type of error that was
encountered.

```
Error no           | Value | Comment
---------------------------------------------------------
```

```
Auth failure        |     0 | Authentication failure
Invalid TS          |     1 | Invalid timestamp
```

```
       Invalid PRF         |      2 | PRF function not supported
       Invalid MAC         |      3 | MAC algorithm not supported
       Invalid EA          |      4 | Encryption algorithm not supported
       Invalid HA          |      5 | Hash function not supported
       Invalid DH          |      6 | DH group not supported
       Invalid ID          |      7 | ID not supported
       Invalid Cert        |      8 | Certificate not supported
       Invalid SP          |      9 | SP type not supported
       Invalid SPpar       |     10 | SP parameters not supported
       Invalid DT          |     11 | not supported Data type
       Unspecified error   |     12 | an unspecified error occurred
```

   Table 6.12

6.13. Key data sub-payload

   The Key data payload contains key material, e.g. TGKs. The Key data
   payloads are never included in clear, but as an encrypted part of the
   Key data transport payload.

   Note that a Key data transport payload can contain multiple Key data
   sub-payloads.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  Next Payload ! Type  ! KV    ! Key data len                 !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                         Key data                             ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Salt len (optional)          ! Salt data (optional)          ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                      KV data (optional)                      ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * Next payload (8 bits): identifies the payload that is added after
   this payload. See Section 6.1 for values.

   * Type (4 bits): indicates the type of the key included in the
   payload.

```
     Type      | Value
     -----------------
     TGK       |     0
     TGK+SALT  |     1
     TEK       |     2
     TEK+SALT  |     3
```

   Table 6.13.a

Note that the possibility to include a TEK (instead of using the TGK)
is provided. When sent directly, the TEK can generally not be shared
between more than one Crypto Session (unless the Security protocol
allows for this, e.g. [SRTP]). The recommended use of sending a TEK
instead of a TGK is when pre-encrypted material exists and therefore,
the TEK must be known in advance.

* KV (4 bits): indicates the type of key validity period specified.
This may be done by using an SPI (or MKI in the case of [SRTP]) or by
providing an interval in which the key is valid (e.g., in the latter
case, for SRTP this will be the index range where the key is valid).

```
   KV               | Value | Comments
   ------------------------------------------
   Null             |     0 | No specific usage rule (e.g. a TEK
                    |       | that has no specific lifetime)
   SPI              |     1 | The key is associated with the SPI/MKI
   Interval         |     2 | The key has a start and expiration time
                    |       | (e.g. an SRTP TEK)
```

   Table 6.13.b

Note that when NULL is specified, any SPI or Interval is valid. For
an Interval this means that the key is valid from the first observed
sequence number until the key is replaced (or the security protocol
is shutdown).

* Key data len (16 bits): the length of the Key data field (in
bytes). Note that the sum of the overall length of all the Key data
payloads contained in a single Key data transport payload (KEMAC)
MUST be such that the KEMAC payload does not exceed a length of 2^{16}
bytes (total length of KEMAC, see Section 6.2).

* Key data (variable length): The TGK or TEK data.

* Salt len (16 bits): The salt key length in bytes. Note that this
field is only included if the salt is specified in the Type-field.

* Salt data (variable length): The salt key data. Note that this
field is only included if the salt is specified in the Type-field.
(For SRTP, this is the so-called master salt.)

* KV data (variable length): This includes either the SPI or an
interval (see Section 6.14). If KV is NULL, this field is not
included.

6.14. Key validity data

The Key validity data is not a standalone payload, but part of either
the Key data payload (see Section 6.13) or the DH payload (see

Section 6.4). The Key validity data gives a guideline of when the key
should be used. There are two KV types defined (see Section 6.13),
SPI/MKI (SPI) or a lifetime range (interval).

SPI/MKI
```
                           1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! SPI Length    ! SPI                                          ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* SPI Length (8 bits): the length of the SPI (or MKI) in bytes.

* SPI (variable length): the SPI (or MKI) value.

Interval
```
                           1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! VF Length     ! Valid From                                   ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! VT Length     ! Valid To (expires)                           ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* VF Length (8 bits): length of the Valid From field in bytes.

* Valid From (variable length): Sequence number, index, timestamp, or
other start value that the security protocol uses to identify the
start position of the key usage.

* VT Length (8 bits): length of the Valid To field in bytes.

* Valid To (variable length): sequence number, index, timestamp, or
other expiration value that the security protocol can use to identify
the expiration of the key usage.

Note that for SRTP usage, the key validity period for a TGK/TEK
should be specified with either an interval, where the VF/VT Length
is equal to 6 bytes (i.e., the size of the index), or with an MKI. It
is RECOMMENDED that if more than one SRTP stream is sharing the same
keys and key update/re-keying is desired, this is handled using MKI
rather than the From-To method.

6.15. General Extension Payload

The General extensions payload is included to allow possible
extensions to MIKEY without the need to define a complete new payload
each time. This payload can be used in any MIKEY message and is part
of the authenticated/signed data part.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next payload  ! Type          ! Length                        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Data                                                          ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Next payload (8 bits): identifies the payload that is added after
this payload.

* Type (8 bits): identifies the type of the general payload.

```
  Type       | Value | Comments
  ----------------------------------------
  Vendor ID  |     0 | Vendor specific byte string
  SDP IDs    |     1 | List of SDP key mgmt IDs (allocated for use in
                       [KMASDP])
```

   Table 6.15

* Length (16 bits): the length in bytes of the Data field.

* Data (variable length): the general payload data.

7. Transport protocols

MIKEY MAY be integrated within session establishment protocols.
Currently integration of MIKEY within SIP/SDP and RTSP is defined in
[KMASDP]. MIKEY MAY use other transport, in which case it has to be
defined how MIKEY is transported over such transport protocol.

8. Groups

What has been discussed up to now is not limited to single peer-to-
peer communication (except for the DH method), but can be used to
distribute group keys for small-size interactive groups and simple
one-to-many scenarios. Section 2.1. describes the scenarios in the
focus of MIKEY. This section describes how MIKEY is used in a group
scenario (though, see also Section 4.3 for issues related to
authorization).

8.1. Simple one-to-many

```
                    ++++
                    |S |
                    |  |
                    ++++
                      |
              --------+------------- - -
              |       |       |
              v       v       v
            ++++     ++++    ++++
            |A |     |B |    |C |
            |  |     |  |    |  |
            ++++     ++++    ++++
```
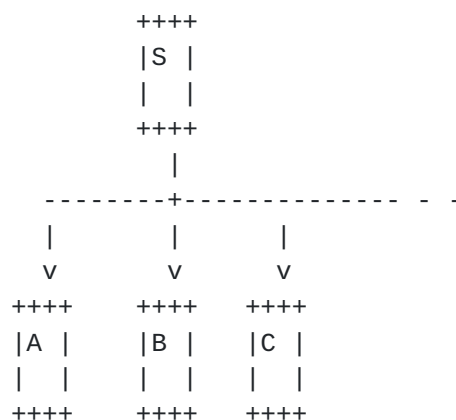
Figure 8.1. Simple one-to-many scenario.

In the simple one-to-many scenario, a server is streaming to a small
group of clients. RTSP or SIP is used for the registration and the
key management set up. The streaming server acts as the Initiator of
MIKEY. In this scenario the pre-shared key or public key transport
mechanism will be appropriate to use to transport the same TGK to all
the clients (which will result in common TEKs for the group).

Note, if the same TGK/TEK(s) should be used by all the group members,
the streaming server MUST specify the same CSB_ID and CS_ID(s) for
the session to all the group members.

As the communication may be performed using multicast, the members
need a common security policy if they want to be part of the group.
This limits the possibility for negotiation.

Furthermore, the Initiator should carefully consider whether to
request the verification message in reply from each receiver, as this
may result in a certain load for the Initiator itself, as the group
size increases.

8.2. Small-size interactive group

As described in the overview section, for small-size interactive
groups, one may expect that each client will be in charge for setting
up the security for its outgoing streams. In these scenarios, the
pre-shared key or the public-key transport method is used.

```
                    ++++           ++++
                    |A | ------->  |B |
                    |  | <-------  |  |
                    ++++           ++++
                     ^ |           | ^
                     | |           | |
                     | |    ++++   | |
                     | --->|C  |<--- |
                     ------|   |------
                           ++++
```
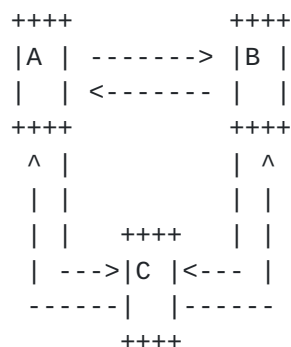
Figure 8.2. Small-size group without centralized controller.

One scenario may then be that the client sets up a three-part call,
using SIP. Due to the small size of the group, unicast SRTP is used
between the clients. Each client sets up the security for its
outgoing stream(s) to the others.

As for the simple one-to-many case, the streaming client specifies
the same CSB_ID and CS_ID(s) for its outgoing sessions if the same
TGK/TEK(s) is used for all the group members.

9. Security Considerations

9.1. General

Key management protocols based on timestamps/counters and one-
roundtrip key transport have previously been standardized in e.g.,
ISO [ISO1, ISO2]. The general security of these types of protocols
can be found in various literature and articles, c.f. [HAC, AKE,
LOA].

No chain is stronger than its weakest link. If a given level of
protection is wanted, then the cryptographic functions protecting the
keys during transport/exchange MUST offer a security at least
corresponding to that level.

For instance, if a security against attacks with complexity 2^{96} is
wanted, then one should choose a secure symmetric cipher supporting
at least 96 bit keys (128 bits may be a practical choice) for the
actual media protection, and a key transport mechanism that provides
equivalent protection, e.g. MIKEY's pre-shared key transport with 128
bit TGK, or, RSA with 1024 bit keys (which according to [LV]
corresponds to the desired 96 bit level, with some margin).

In summary, key size for the key-exchange mechanism MUST be weighed
against the size of the exchanged TGK so that it offers at least the

required level. For efficiency reasons, one SHOULD also avoid a
security overkill, e.g. by not using a public key transport with

public keys giving a security level that is orders of magnitude
higher than length of the transported TGK. We refer to [LV] for
concrete key size recommendations.

Moreover, if the TGKs are not random (or pseudo-random), a brute
force search may be facilitated, again lowering the effective key
size. Therefore, care MUST be taken when designing the (pseudo-)
random generators for TGK generation, see [FIPS][RAND].

For the selection of the hash function, SHA-1 with 160-bit output is
the default one. In general, hash sizes should be twice the "security
level", indicating that SHA-1-256, [SHA256], should be used for the
default 128-bit level. However, due to the real-time aspects in the
scenarios we are treating, hash size slightly below 256 are
acceptable as the normal "existential" collision probabilities would
be of secondary importance.

In a Crypto Session Bundle, the Crypto Sessions can share the same
TGK as discussed earlier. From a security point of view, the
criterion to be satisfied in case the TGK is shared, is that the
encryption of the individual Crypto Sessions are performed
"independently". In MIKEY this is accomplished by having unique
Crypto Session identifiers (see also Section 4.1) and a TEK
derivation method that provides cryptographically independent TEKs to
distinct Crypto Sessions (within the Crypto Session Bundle),
regardless of the security protocol used.

Specifically, the key derivations, as specified in Section 4.1, are
implemented by a pseudo-random function. The one used here is a
simplified version of that used in TLS [TLS]. Here, only one single
hash function is used, whereas TLS uses two different functions. This
choice is motivated by the high confidence in the SHA-1 hash
function, and by efficiency and simplicity of design (complexity does
not imply security). Indeed, as shown in [DBJ], if one of the two
hashes is severely broken, the TLS PRF is actually less secure than
if a single hash had been used on the whole key, as is done in MIKEY.

In the pre-shared key and public-key schemes, the TGK is generated by
a single party (Initiator). This makes MIKEY somewhat more sensitive
if the Initiator uses a bad random number generator. It should also
be noted that neither the pre-shared nor the public-key scheme
provides perfect forward secrecy. If mutual contribution or perfect
forward secrecy is desired, the Diffie-Hellman method is to be used.
Authentication (e.g. signatures) in the Diffie-Hellman method is
required to prevent man-in-the-middle attacks.

Forward/backward security: if the TGK is exposed, all TEKs generated
from it are compromised. However, under the assumption that the

derivation function is a pseudo-random function, disclosure of an
individual TEK does not compromise other (previous or later) TEKs
derived from the same TGK. The Diffie-Hellman mode can be considered

by cautious users as it is the only one that supports so called perfect forward secrecy (PFS). This is in contrast to a compromise of the pre-shared key (or the secret key of the public key mode), where future sessions and recorded session from the past are then also compromised.

The use of random nonces (RANDs) in the key derivation is of utmost importance to counter off-line pre-computation attacks. Note however that update messages re-use the old RAND. This means that the total effective key entropy (relative to pre-computation attacks) for k consecutive key updates, assuming the TGKs and RAND are each n bits long, is about L = n*(k+1)/2 bits, compared to the theoretical maximum of n*k bits. In other words, a 2^L work effort MAY enable an attacker to get all k n-bit keys, which is better than brute force (except when k = 1). While this might seem as a defect, first note that for proper choice of n, the 2^L complexity of the attack is way out of reach. Moreover, the fact that more than one key can be compromised in a single attack is inherent to the key exchange problematic. Consider for instance a user who, using say a fixed 1024-bit RSA key, exchanges keys and communicates during one or two years lifetime of the public key. Breaking this single RSA key will enable access to all exchanged keys and consequently the entire communication of that user over the whole period.

All the pre-defined transforms in MIKEY use state-of-the-art algorithms that have undergone large amounts of public evaluation. One of the reasons to use AES-CM from SRTP [SRTP] is to have the possibility to limit the overall number of different encryption modes and algorithms, at the same time that it offers a high level of security.

9.2. Key lifetime

Even if the lifetime of a TGK (or TEK) is not specified, it MUST be taken into account that the encryption transform in the underlying security protocol can in some way degenerate after a certain amount of encrypted data. It is not possible to here state general key lifetime bounds, universally applicable; each security protocol should define such maximum amount and trigger a re-keying procedure before the "exhaustion" of the key. E.g., according to SRTP [SRTP] the TEK, together with the corresponding TGK, MUST be changed at least every 2^48 SRTP packet.

Still, the following can be said as a rule of thumb. If the security protocol uses an "ideal" b-bit block cipher (in CBC mode, counter mode, or a feedback mode, e.g. OFB, with full b-bit feedback), degenerate behavior in the crypto stream, possibly useful for an attacker, is (with constant probability) expected to occur after a

total of roughly 2^(b/2) encrypted b-bit blocks (using random IVs).
For security margin, re-keying MUST be triggered well in advance
compared to the above bound. See [BDJR] for more details.

For use of a dedicated stream cipher, we refer to the analysis and documentation of said cipher in each specific case.

9.3. Timestamps

The use of timestamps instead of challenge-response requires the systems to have synchronized clocks. Of course, if two clients are not synchronized, they will have difficulties with setting up the security. The current timestamp based solution has been selected to allow a maximum of one roundtrip (i.e., two messages), but still provide a reasonable replay protection. A (secure) challenge-response based version would require at least three messages. For a detailed description of the timestamp and replay handling in MIKEY, see Section 5.4.

Practical experiences of Kerberos and other timestamp-based systems indicate that it is not always necessary to synchronize the terminals over the network. Manual configuration could be a feasible alternative in many cases (especially in scenarios where the degree of looseness is high). However, the choice must be carefully based with respect to the usage scenario.

9.4. Identity protection

User privacy is a complex matter that to some extent can be enforced by cryptographic mechanisms, but also requires policy enforcement and various other functionalities. One particular facet of privacy is user identity protection. However, identity protection was not a main design goal for MIKEY. Such feature will add more complexity to the protocol and was therefore chosen not to be included. As MIKEY is anyway proposed to be transported over e.g. SIP, the identity may be exposed by this. However, if the transporting protocol is secured and also provides identity protection, MIKEY might inherit the same feature. How this should be done is for future study.

9.5. Denial of Service

This protocol is resistant to Denial of Service attacks in the sense that a Responder does not construct any state (at the key management protocol level) before it has authenticated the Initiator. However, this protocol, like many others, is open to attacks that use spoofed IP addresses to create a large number of fake requests. This may e.g., be solved by letting the protocol transporting MIKEY do an IP address validity test. For example, the SIP protocol can provide this using the anonymous authentication challenge mechanism (specified in Section 22.1 of [SIP]).

As also discussed in Section 5.4, the tradeoff between time

synchronization and the size of the replay cache, may be affected in
case of e.g., a flooding type of DoS attack. However, if the

recommendations of using a dynamic size of the replay cache are
followed, it is believed that the client will in most cases be able
to handle the replay cache. Of course, as the replay cache decreases
in size, the required time synchronization is more restricted.
However, a bigger problem during such attack would probably be to
process the messages (e.g., verify signatures/MACs), due to the
computational workload this implies.

9.6. Session establishment

It should be noted that if the session establishment protocol is
insecure there may be attacks on this that will have indirect
security implications on the secured media streams. This however only
applies to groups (and is not specific to MIKEY). The threat is that
one group member may re-direct a stream from one group member to
another. This will have the same implication as when a member tries
to impersonate another member, e.g. by changing its IP address. If
this is seen as a problem, it is RECOMMENDED that a Source Origin
Authentication (SOA) scheme (e.g., digital signatures) is applied to
the security protocol.

Re-direction of streams can of course be done even if it is not a
group. However, the effect will not be the same compared to a group
where impersonation can be done if SOA is not used. Instead, re-
direction will only deny the receiver the possibility to receive (or
just delay) the data.

10. IANA considerations

This document defines several new name spaces associated with the
MIKEY payloads. This section summarizes the name spaces for which
IANA is requested to manage the allocation of values.
IANA is requested to record the pre-defined values defined in the
given sections for each name space. IANA is also requested to manage
the definition of additional values in the future. Unless explicitly
stated otherwise, values in the range 0-240 for each name space
SHOULD be approved by the process of IETF consensus and values in the
range 241-255 are reserved for Private Use, according to [RFC2434].

The name spaces for the following fields in the Common header payload
(from Section 6.1) are requested to be managed by IANA (in bracket is
the reference to the table with initial registered values):

* version

* data type (Table 6.1.a)

* Next payload (Table 6.1.b)

* PRF func (Table 6.1.c). This name space is between 0-127 where
values between 0-111 should be approved by the process of IETF
consensus and values between 112-127 are reserved for Private Use.

* CS ID map type (Table 6.1.d)

The name spaces for the following fields in the Key data transport
payload (from Section 6.2) are requested to be managed by IANA:

* Encr alg (Table 6.2.a)

* MAC alg (Table 6.2.b)

The name spaces for the following fields in the Envelope data payload
(from Section 6.3) are requested to be managed by IANA:

* C (Table 6.3)

The name spaces for the following fields in the DH data payload (from
Section 6.4) are requested to be managed by IANA:

* DH-Group (Table 6.4)

The name spaces for the following fields in the Signature payload
(from Section 6.5) are requested to be managed by IANA:

* S type (Table 6.5)

The name spaces for the following fields in the Timestamp payload
(from Section 6.6) are requested to be managed by IANA:

* TS type (Table 6.6)

The name spaces for the following fields in the ID payload and the
Certificate payload (from Section 6.7) are requested to be managed by
IANA:

* ID type (Table 6.7.a)

* Cert type (Table 6.7.b)

The name spaces for the following fields in the Cert hash payload
(from Section 6.8) are requested to be managed by IANA:

* Hash func (Table 6.8)

The name spaces for the following fields in the Security policy
payload (from Section 6.10) are requested to be managed by IANA:

* Prot type (Table 6.10)

For each security protocol that uses MIKEY, a set of unique
parameters MAY be registered.

From Section 6.10.1.

* SRTP Type (Table 6.10.1.a)

* SRTP encr alg (Table 6.10.1.b)

* SRTP auth alg (Table 6.10.1.c)

* SRTP PRF (Table 6.10.1.d)

* FEC order (Table 6.10.1.e)

The name spaces for the following fields in the Error payload (from
Section 6.12) are requested to be managed by IANA:

* Error no (Table 6.12)

The name spaces for the following fields in the Key data payload
(from Section 6.13) are requested to be managed by IANA:

* Type (Table 6.13.a). This name space is between 0-16 which should
be approved by the process of IETF consensus.

* KV (Table 6.13.b). This name space is between 0-16 which should be
approved by the process of IETF consensus.

The name spaces for the following fields in the General Extensions
payload (from Section 6.15) are requested to be managed by IANA:

* Type (Table 6.15).

10.1 MIME Registration

This section gives instructions to IANA to register the
application/mikey MIME media type. This registration is as follows:

```
 MIME media type name              : application
 MIME subtype name                 : mikey
 Required parameters               : none
 Optional parameters               : version
           version: The MIKEY version number of the enclosed message
              (e.g., 1). If not present, the version defaults to 1.
 Encoding Considerations           : binary, base64 encoded
 Security Considerations           : see section 9 in this memo
 Interoperability considerations   : none
 Published specification           : this memo
```

11. Acknowledgments

The authors would like to thank Mark Baugher, Ran Canetti, Martin
Euchner, Steffen Fries, Peter Barany, Russ Housley, Pasi Ahonen (with
his group), Rolf Blom, Magnus Westerlund, Johan Bilien, Jon-Olov
Vatn, and Erik Eliasson for their valuable feedback.

12. 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

13. References

13.1. Normative References

[AES] Advanced Encryption Standard (AES), Federal Information
Processing Standard Publications (FIPS PUBS) 197, November 2001.

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

[NAI] Aboba, B. and Beadles, M., "The Network Access Identifier",
IETF, RFC 2486, January 1999.

   [OAKLEY] Orman, H., "The Oakley Key Determination Protocol", RFC
   2412, November 1998.

[PSS] PKCS #1 v2.1 - RSA Cryptography Standard, RSA Laboratories,
June 14, 2002, www.rsalabs.com

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

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

[RSA] Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining
Digital Signatures and Public-Key Cryptosystems". Communications of
the ACM. Vol.21. No.2. pp.120-126. 1978.

[SHA-1] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.
http://csrc.nist.gov/fips/fip180-1.ps

[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 WG).

[URI] Berners-Lee. T., Fielding, R., Masinter, L., "Uniform Resource
Identifiers (URI): Generic Syntax", IETF, RFC 2396.

[X.509] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet
X.509 Public Key Infrastructure Certificate and Certificate
Revocation List (CRL) Profile", IETF, RFC 3280.

[AESKW] Schaad, J., Housley R., "Advanced Encryption Standard (AES)
Key Wrap Algorithm", IETF, RFC 3394.

13.2. Informative References

[AKE] Canetti, R. and Krawczyk, H., "Analysis of Key-Exchange
Protocols and their use for Building Secure Channels", Eurocrypt
2001, LNCS 2054, pp. 453-474, 2001.

[BDJR] Bellare, M., Desai, A., Jokipii, E., and Rogaway, P., "A
Concrete Analysis of Symmetric Encryption: Analysis of the DES Modes
of Operation", in Proceedings of the 38th Symposium on Foundations of
Computer Science, IEEE, 1997, pp. 394-403.

[BMGL] Hastad, J. and Naslund, M.: "Practical Construction and
Analysis of Pseduo-randomness Primitives", Proceedings of Asiacrypt
'01, Lecture Notes in Computer Science vol 2248, pp. 442-459.

[DBJ] Johnson, D.B., "Theoretical Security Concerns with TLS use of
MD5", Contribution to ANSI X9F1 WG, 2001.

[FIPS] "Security Requirements for Cryptographic Modules", Federal
Information Processing Standard Publications (FIPS PUBS) 140-2,
December 2002.

[GKMARCH] Baugher, M., Canetti, R., Dondeti, L., and Lindholm, F.,
"Group Key Management Architecture", Internet Draft, Work in Progress
(MSEC WG).

[GDOI] Baugher, M., Hardjono, T., Harney, H., Weis, B., "The Group
Domain of Interpretation", Internet Draft, Work in Progress (MSEC
WG).

[GSAKMP] Harney, H., Colegrove, A., Harder, E., Meth, U., Fleischer,
R., "Group Secure Association Key Management Protocol", Internet
Draft, Work in Progress (MSEC WG).

[HAC] Menezes, A., van Oorschot, P., and Vanstone, S., "Handbook of
Applied Cryptography", CRC press, 1996.

[IKE] Harkins, D. and Carrel, D., "The Internet Key Exchange (IKE)",
RFC 2409, November 1998.

[ISO1] ISO/IEC 9798-3: 1997, Information technology - Security
techniques - Entity authentication - Part 3: Mechanisms using digital
signature techniques.

[ISO2] ISO/IEC 11770-3: 1997, Information technology - Security
techniques - Key management - Part 3: Mechanisms using digital
signature techniques.

[ISO3] ISO/IEC 18014 Information technology - Security techniques -
Time-stamping services, Part 1-3.

[KMASDP] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and
Norrman, K., "Key Management Extensions for SDP and RTSP", Internet
Draft, Work in Progress (MMUSIC WG).

[LOA] Burrows, Abadi, and Needham, "A logic of authentication", ACM
Transactions on Computer Systems 8 No.1 (Feb. 1990), 18-36.

[LV] Lenstra, A. K., and Verheul, E. R., "Suggesting Key Sizes for
Cryptosystems", http://www.cryptosavvy.com/suggestions.htm

[NTP] Mills, D., "Network Time Protocol (Version 3) specification,
implementation and analysis", RFC 1305, March 1992.

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams
C., "X.509 Internet Public Key Infrastructure Online Certificate
Status Protocol - OCSP", IETF, RFC 2560.

[RAND] Eastlake, D., Schiller, J., and Crocker, S., "Randomness
Requirements for Security", RFC 1750, December 1994.

[RTSP] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time
Streaming Protocol (RTSP)", RFC 2326, April 1998.

[SDP] Handley, M., Jacobson, V., and Perkins, C., "SDP: Session
Description Protocol", Internet Draft, IETF, Work in progress
(MMUSIC), draft-ietf-mmusic-sdp-new-15.txt.

[SHA256] NIST, "Description of SHA-256, SHA-384, and SHA-512",
http://csrc.nist.gov/encryption/shs/sha256-384-512.pdf

[SIP] Rosenberg, J. et al, "SIP: Session Initiation Protocol", IETF,
RFC3261.

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

Appendix A. - MIKEY - SRTP relation

The terminology in MIKEY differs from the one used in SRTP as MIKEY
needs to be more general, nor is tight to SRTP only. Therefore it
might be hard to see the relations between keys and parameters
generated in MIKEY and the ones used by SRTP. This section provides
some hints on their relation.

```
MIKEY             | SRTP
-------------------------------------------------
Crypto Session    | SRTP stream (typically with related SRTCP stream)
Data SA           | input to SRTP's crypto context
TEK               | SRTP master key
```

The Data SA is built up by a TEK and the security policy exchanged.
SRTP may use a MKI to index the TEK, or TGK (the TEK is then derived
from the TGK that is associated with the corresponding MKI), see
below.

A.1 MIKEY-SRTP interactions

In the following, we give a brief outline of the interface between
SRTP and MIKEY and the processing that takes place. We describe SRTP
receiver side only, the sender side will require analogous
interfacing.

1. When an SRTP packet arrives at the receiver and is processed, the
triple <SSRC, destination address, destination port> is extracted
from the packet and used to retrieve the correct SRTP crypto context,

hence the Data SA. (The actual retrieval can e.g. be done by an
explicit request from the SRTP implementation to MIKEY, or, by the

SRTP implementation accessing a "data base", maintained by MIKEY. The
application will typically decide which implementation is preferred.)

2. If an MKI is present in the SRTP packet, it is used to point to
the correct key within the SA. (Alternatively, if SRTPÆs <From, To>
feature is used, the ROC||SEQ of the packet is used to determine the
correct key.)

3. Depending on whether the key sent in MIKEY (as obtained in step 2)
was a TEK or a TGK, there are now two cases.

   - If the key obtained in step 2 is the TEK itself, it is used
      directly by STRP as a master key.

   - If the key instead is a TGK, the mapping with the CS_ID (internal
      to MIKEY, Section 6.1.1) allows MIKEY to compute the correct TEK
      from the TGK as described in Section 4.1 before SRTP uses it.

If multiple TGKs (or TEKs) are sent, it is RECOMMENDED to associate
each TGK (or TEK) to a distinct MKI. It is RECOMMENDED to limit the
use of <From, To> in this scenario to very simple cases, e.g. one
stream only.

Besides the actual master key, other information in the Data SA (e.g.
transform identifiers) will of course also be communicated from MIKEY
to SRTP.

IPR Notices

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

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

Copyright Notice

This Internet-Draft expires in June 2004.