Internet Engineering Task Force (IETF)
Request for Comments: 6043
Category: Informational
ISSN: 2070-1721

J. Mattsson
Ericsson
T. Tian
ZTE

March 2011

MIKEY-TICKET: Ticket-Based Modes of Key Distribution in Multimedia Internet KEYing (MIKEY)

Abstract

The Multimedia Internet KEYing (MIKEY) specification describes a key management scheme for real-time applications. In this document, we note that the currently defined MIKEY modes are insufficient to address deployment scenarios built around a centralized key management service. Interest in such deployments is increasing. Therefore, a set of new MIKEY modes that work well in such scenarios are defined. The new modes use a trusted key management service and a ticket concept, similar to that in Kerberos. The new modes also support features used by many existing applications, where the exact identity of the other endpoint may not be known at the start of the communication session.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Key management systems are either based on negotiation and exchange directly between peers (e.g., Diffie-Hellman-based schemes), pre-distribution of user credentials (shared secrets/certificates), or availability of a trusted Key Management Service (KMS). The modes described in the Multimedia Internet KEYing (MIKEY) specification [RFC3830] and its extensions [RFC4650] [RFC4738] are all variants of the first two alternatives.

In security systems serving a large number of users, a solution based on a key management service is often preferred. With such a service in place, there is no need to pre-distribute credentials that directly can be used to establish security associations between peers for protected communication, as users can request such credentials when needed. Solutions based on a trusted key management service also scale well when the number of users grows.

This document introduces a set of new MIKEY modes that go under the common name MIKEY-TICKET. The new modes support a ticket concept, similar to that in Kerberos [RFC4120], which is used to identify and deliver keys. A high-level outline of MIKEY-TICKET as defined herein is that the Initiator requests keys and a ticket from the KMS and sends the ticket to the Responder. The ticket contains a reference to the keys, or the enveloped keys. The Responder then sends the ticket to the KMS, which returns the appropriate keys.

MIKEY-TICKET is primarily designed to be used for media plane security in the 3rd Generation Partnership Project (3GPP) IP Multimedia Subsystem (IMS) [3GPP.33.328]. This implies that some extensions to the basic Kerberos concept are needed. For instance, the Initiator may not always know the exact identity of the Responder when the communication with the key management server is initiated.

This document defines a signaling framework enabling peers to request, transfer, and resolve various Ticket Types using a key management service. A default Ticket Type is also defined. To allow the use of 256-bit keys for users with high security requirements, additional encryption, authentication, and pseudorandom functions are defined. And to eliminate the limitations with the existing SRTP-ID map type, a new CS ID map type called GENERIC-ID is defined.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Definitions of terms and notation will, unless otherwise stated, be as defined in [RFC3830].

2.1. Definitions and Notation

Forking: The delivery of a request to multiple endpoints (multiple devices owned by a single user or multiple users).

Key forking: When used in conjunction with forking, key forking refers to the process of modifying keys, making them cryptographically unique for each responder targeted by the forking.

(Media) session: The communication session intended to be secured by the MIKEY-TICKET provided key(s).

Session information: Information related to the security protocols used to protect the media session: keys, salts, algorithms, etc.

Ticket: A Kerberos-like object used to identify and deliver keys over an untrusted network.

Ticket Data: Ticket part with information intended only for the party that resolves the ticket (e.g., keys).

Ticket Request: Exchange used by the Initiator to request keys and a ticket from a trusted KMS.

Ticket Transfer: Exchange used to transfer the ticket as well as session information from the Initiator to the Responder.

Ticket Resolve: Exchange used by the Responder to request the KMS to return the keys encoded in a ticket.

Ticket Policy: Policy for ticket generation and resolution, authorized applications, key derivation, etc.

Ticket Type: Defines ticket format and processing. May further have subtype and version.

Solid arrows (----->) indicate mandatory messages.
Dashed arrows (- - ->) indicate optional messages.

E(k, p)	Encryption of p with the key k
PKx	Public Key of entity x
k'	The forked key k
[p]	p is optional
{p}	Zero or more occurrences of p
(p)	One or more occurrences of p

||	Concatenation
|	OR (selection operator)

2.2. Abbreviations

3GPP:	3rd Generation Partnership Project
AAA:	Authentication, Authorization, and Accounting
ACL:	Access Control List
AES:	Advanced Encryption Standard
CA:	Certification Authority
CS:	Crypto Session
CSB:	Crypto Session Bundle
IMS:	IP Multimedia Subsystem
GTGK:	Group TGK
HMAC:	Hash-based Message Authentication Code
KMS:	Key Management Service
MAC:	Message Authentication Code
MIKEY:	Multimedia Internet KEYing
NSPS:	National Security and Public Safety
MKI:	Master Key Identifier
MPK:	MIKEY Protection Key
NTP:	Network Time Protocol
PET:	Privacy Enhancing Technologies
PK:	Public Key
PRF:	Pseudorandom Function
PRNG:	Pseudorandom Number Generator
PSK:	Pre-Shared Key
RTSP:	Real Time Streaming Protocol
SDP:	Session Description Protocol
SHA:	Secure Hash Algorithm
SIP:	Session Initiation Protocol
SPI:	Security Parameters Index
SRTP:	Secure Realtime Transport Protocol
TEK:	Traffic Encryption Key
TGK:	TEK Generation Key
TPK:	Ticket Protection Key
UTC:	Coordinated Universal Time

2.3. Payloads

CERTx:	Certificate of entity x
CHASH:	Hash of the certificate used
HDR:	Common Header payload
ID:	Identity payload
IDRx:	Identifier for entity x
IDRpsk:	Identifier for pre-shared key
IDRapp:	Identifier for application/service
KEMAC:	Key data transport payload

PKE:	Encrypted envelope key
RAND:	RAND payload
RANDRx:	Random value generated by entity x
SIGNx:	Signature created using entity x's private key
SP:	Security Policy payload
T:	Timestamp payload
TRy:	Timestamp payload with role indicator y
THDR:	Ticket Header payload
TICKET:	Ticket payload
TP:	Ticket Policy payload
V:	Verification payload

where
x is in the set {i, r, kms} (Initiator, Responder, KMS) and
y is in the set {i, s, e, r} (time of Issue, Start time, End time, Rekeying interval).

The IDR, RANDR, TR, TICKET, and TP payloads are defined in Section 6. Note that in [RFC3830], there is defined both a V payload (carrying the authentication tag) and a V flag in the HDR payload (indicating whether or not a response message is expected).

3. Design Considerations

As mentioned in the introduction, none of the previously defined MIKEY modes are based on a KMS. The pre-shared key method and the public-key encryption method defined in [RFC3830] are examples of systems based on pre-distribution of user credentials. The Diffie-Hellman method [RFC3830] is an example of a system based on negotiation and exchange directly between peers.

In some situations, a request may be delivered to multiple endpoints. The endpoints may be multiple devices owned by a single user (e.g., mobile phone, fixed phone, and computer), or multiple users (e.g., IT-support@example.com, a group of users where only one is supposed to answer). In the following, the term "forking" will be used to describe all such cases. One example of delivery to multiple endpoints is forking and retargeting in SIP [RFC3261]. To prevent any form of eavesdropping, only the endpoint that answers should get access to the session keys. The naive application of [RFC3830] where all endpoints share the same pre-shared/private key is not secure when it comes to forking, as all endpoints get access to the session keys. Conversely, having per-user unique pre-shared keys/certificates creates more fundamental problems with forking, as the initiator does not know which pre-shared key/certificate to use at session initiation. SIP-signaled media protection is described in [RFC5479] and the applicability of different MIKEY modes is discussed in [RFC5197].

In security systems serving a large number of users, a solution based on a key management service is often preferred. With such a service in place, there is no need to pre-distribute credentials that directly can be used to establish security associations between peers for protected communication, as users can request such credentials when needed. In many applications, e.g., National Security and Public Safety (NSPS), the controlling organization wants to enforce policies on the use of keys. A trusted KMS fits these applications well, as it makes it easier to enforce policies centrally. Solutions based on a trusted KMS also scale well when the number of users grows. A KMS based on symmetric keys has particular advantages, as symmetric key algorithms are generally much less computationally intensive than asymmetric key algorithms.

Systems based on a KMS require a signaling mechanism that allows peers to retrieve other peers' credentials. A convenient way to implement such a signaling scheme is to use a ticket concept, similar to that in Kerberos [RFC4120] to identify and deliver keys. The ticket can be forwarded in the signaling associated with the session setup. The initiator requests a ticket from the KMS and sends the ticket to the responder. The responder forwards the ticket to the KMS, which returns the corresponding keys.

It should be noted that Kerberos does not require that the responder also contact the KMS. However, in order to support also the aforementioned forking scenarios, it becomes necessary that the ticket is not bound to the exact identity (or credentials) of the responder until the final responder becomes fully determined. Group and forking communication scenarios can also be improved from access control point of view if authorization to access the keys can be enforced with higher granularity at the responder side. The mechanism specified in this document is useful for any system where the initial message may be transferred to arbitrarily many potential responders and where the set of responders may change at any time. In addition to being able to meet the requirements just described, the mechanism specified in this document also supports group key management.

The ticket can contain a reference to keys held by the key management system or it can hold the keys itself. In the latter case, the ticket needs to be confidentiality and integrity protected (enveloped). In the following, the term "encoded keys" will be used to describe both cases as well as keys derived from such keys.

By using different Ticket Types and ticket policies, some allowing the initiator or responder to create or resolve the tickets without assistance from the KMS, a wide range of different security levels and use cases can be supported. This has a number of advantages, as it offers a framework that is flexible enough to satisfy users with a broad range of security and functional needs.

The use of a ticket-based system may also help in the handling of keys for deferred delivery of end-to-end protected content to currently offline users. Such scenarios exclude all key management schemes that are based on some type of direct online negotiation between peers (e.g., Diffie-Hellman-based schemes) as the responder cannot rely on contacting the initiator to get access to keys.

At the same time, it is also important to be aware that (centralized) key management services may introduce a single point of (security) failure. The security requirements on the implementation and protection of the KMS may therefore, in high-security applications, be more or less equivalent to the requirements of an AAA (Authentication, Authorization, and Accounting) server or a Certification Authority (CA).

4. MIKEY-TICKET

4.1. Overview

All previously defined MIKEY modes consist of a single (or half) round trip between two peers. MIKEY-TICKET differs from these modes as it consists of up to three different round trips (Ticket Request, Ticket Transfer, and Ticket Resolve) involving three parties (Initiator, Responder, and KMS). Since the number of round trips and order of messages may vary, MIKEY-TICKET is actually the common name for a set of modes, all revolving around a ticket concept. The third party, the KMS, is only involved in some of the MIKEY exchanges and not at all in the resulting secure media session. The Ticket Request and Ticket Resolve exchanges are meant to be used in combination with the Ticket Transfer exchange and not on their own. In Figure 1, the signaling for the full three round-trip MIKEY-TICKET mode is depicted.

```
   +---+                          +-----+                         +---+
   | I |                          | KMS |                         | R |
   +---+                          +-----+                         +---+
             REQUEST_INIT
     -------------------------------->
             REQUEST_RESP
     <--------------------------------
                             TRANSFER_INIT
     ---------------------------------------------------------------->
                                              RESOLVE_INIT
                                     <--------------------------------
                                              RESOLVE_RESP
                                     -------------------------------->
                             TRANSFER_RESP
     <----------------------------------------------------------------
```

Figure 1: Full three round-trip signaling

The Initiator (I) wants to establish a secure media session with the Responder (R). The Initiator and the Responder do not share any credentials; instead, they trust a third party, the KMS, with which they both have or can establish shared credentials. These pre-established trust relations are used to establish a security association between I and R. The assumed trust model is illustrated in Figure 2.

```
      Pre-established trust relation   Pre-established trust relation
     <-------------------------------> <------------------------------->
   +---+                          +-----+                         +---+
   | I |                          | KMS |                         | R |
   +---+                          +-----+                         +---+
     <---------------------------------------------------------------->
                    Security association based on ticket
```

Figure 2: Trust model

Note that rather than a single KMS, multiple KMSs may be involved, e.g., one for the Initiator and one for the Responder; this is discussed in Section 10.

The Initiator requests keys and a ticket (encoding the same keys) from the KMS by sending a REQUEST_INIT message. The REQUEST_INIT message includes session information (e.g., identities of the authorized responders) and is integrity protected by a MAC based on a pre-shared key or by a signature (similar to the pre-shared key and public-key encryption modes in [RFC3830]). If the request is authorized, the KMS generates the requested keys, encodes them in a ticket, and returns the keys and the ticket in a REQUEST_RESP message. The Ticket Request exchange is OPTIONAL (depending on the Ticket Type), and MAY be omitted if the Initiator can create the ticket without assistance from the KMS (see mode 3 of Section 4.1.1).

The Initiator next includes the ticket in a TRANSFER_INIT message, which is sent to the Responder. The TRANSFER_INIT message is protected by a MAC based on an MPK (MIKEY Protection Key) encoded in the ticket. If the Responder finds the Ticket Policy and session security policies acceptable, the Responder forwards the ticket to the KMS. This is done with a RESOLVE_INIT message, which asks the KMS to return the keys encoded in the ticket. The RESOLVE_INIT message is protected by a MAC based on a pre-shared key (between Responder and KMS) or by a signature. The Ticket Resolve exchange is OPTIONAL (depending on the Ticket Policy), and SHOULD only be used when the Responder is unable to resolve the ticket without assistance from the KMS (see mode 2 of Section 4.1.1).

The KMS resolves the ticket. If the Responder is authorized to receive the keys encoded in the ticket, the KMS retrieves the keys and other information. If key forking is used, the keys are modified (bound to the Responder) by the KMS, see Section 5.1.1. The keys and additional information are then sent in a RESOLVE_RESP message to the Responder. The Responder then sends a TRANSFER_RESP message to the Initiator as verification. The TRANSFER_RESP message might include information used for further key derivation.

The use case and signaling described above is the full three round-trip mode, but other modes are allowed, see Section 4.1.1. Pre-encrypted content is discussed in Section 8, group communication is discussed in Section 9, and signaling between different KMSs is discussed in Section 10. An alternative use case is discussed in Appendix B.

The session keys are normally generated/supplied by the KMS (encoded in the ticket), but in certain use cases (see Section 8) the session key may be supplied by the Initiator or Responder (sent in a separate KEMAC protected with keys derived from the MPK).

MIKEY-TICKET offers a framework that is flexible enough to satisfy users with a broad range of security and functional needs. The framework consists of the three exchanges for which different Ticket Types can be defined. The ticket consists of a Ticket Policy as well as Ticket Data. The Ticket Policy contains information intended for all parties involved, whereas the Ticket Data is only intended for the party that resolves the ticket. The Ticket Data could be a reference to information (keys, etc.) stored by the key management service, it could contain all the information itself, or it could be a combination of the two alternatives. The format of the Ticket Data depends on the Ticket Type signaled in the Ticket Policy. The Ticket Data corresponding to the default Ticket Type, called MIKEY base ticket, is defined in Appendix A and requirements regarding new Ticket Types are given in Section 11.

As MIKEY-TICKET is based on [RFC3830], the same terminology, processing, and considerations still apply unless otherwise stated. Just like in [RFC3830], the messages are integrity protected and encryption is only applied to the keys and not to the entire messages.

4.1.1. Modes

Depending on the Ticket Type and the Ticket Policy, some of the exchanges might be optional or not used at all, see Figure 3. If the ticket protection is based on a key known only by the KMS, both the Initiator and the Responder have to contact the KMS to request/ resolve tickets (mode 1). If the key used to protect the ticket is shared between the KMS and the Responder, the Ticket Resolve exchange can be omitted (similar to Kerberos), as the Responder can resolve the ticket without assistance from the KMS (mode 2).

```
     +---+                         +-----+                         +---+
     | I |                         | KMS |                         | R |
     +---+                         +-----+                         +---+
                Ticket Request
   (1) <------------------------------->       Ticket Transfer
       <------------------------------------------------------------->
                                  <------------------------------->
                                               Ticket Resolve
                Ticket Request
   (2) <------------------------------->       Ticket Transfer
       <------------------------------------------------------------->

                               Ticket Transfer
   (3) <------------------------------------------------------------->
                                  <------------------------------->
                                               Ticket Resolve

                               Ticket Transfer
   (4) <------------------------------------------------------------->
```

Figure 3: Modes

If the key protecting the ticket is shared between the Initiator and the KMS, the Ticket Request exchange can be omitted (similar to the Otway-Rees protocol [Otway-Rees]), as the Initiator can create the ticket without assistance from the KMS (mode 3). If the key protecting the ticket is shared between the Initiator and the Responder, both the Ticket Request and Ticket Resolve exchanges can be omitted (mode 4). This can be seen as a variation of the pre-shared key method of [RFC3830] with a mutual key-freshness guarantee.

In modes 1 and 2, the Ticket Request exchange can be omitted if the tickets and the corresponding keys are distributed from the KMS to the Initiator in some other way. In addition, as tickets may be reused (see Section 5.3), a single Ticket Request exchange may be followed by several Ticket Transfer exchanges.

4.2. Exchanges

4.2.1. Ticket Request

This exchange is used by the Initiator to request keys and a ticket from a trusted KMS with which the Initiator has pre-shared credentials. The request contains information (e.g., participant identities, etc.) describing the session the ticket is intended to protect. A full round trip is required for the Initiator to receive the ticket. The initial message REQUEST_INIT comes in two variants. The first variant corresponds to the pre-shared key (PSK) method of [RFC3830].

```
Initiator                                KMS

REQUEST_INIT_PSK =                ---->
HDR, T, RANDRi, [IDRi],
   [IDRkms], TP,                   <----  REQUEST_RESP =
   [IDRpsk], V                            HDR, T, [IDRkms],
                                             TICKET, KEMAC, V
```

The second variant corresponds to the public-key (PK) method of [RFC3830].

```
Initiator                                KMS

REQUEST_INIT_PK =                 ---->
HDR, T, RANDRi, [IDRi],
   {CERTi}, [IDRkms], TP,          <----  REQUEST_RESP =
   [CHASH], PKE, SIGNi                    HDR, T, [IDRkms],
                                             TICKET, KEMAC, V
```

As the REQUEST_INIT message MUST ensure the identity of the Initiator to the KMS, it SHALL be integrity protected by a MAC based on a pre-shared key or by a signature. The response message REQUEST_RESP is the same for the two variants and SHALL be protected using the pre-shared/envelope key indicated in the REQUEST_INIT message.

In addition to the ticket, the Initiator receives keys, which it does not already know. The ticket contains both session information and information needed to resolve the ticket later, see Section 6.10.

4.2.1.1. Common Components of the REQUEST_INIT Messages

The REQUEST_INIT message MUST always include the Header (HDR), Timestamp (T), and RANDRi payloads.

In HDR, the CSB ID (Crypto Session Bundle ID) SHALL be assigned as in [RFC3830]. The V flag MUST be set to '1' but SHALL be ignored by the KMS as a response is MANDATORY. As Crypto Sessions (CSs) SHALL NOT be handled, the #CS MUST be set to '0' and the CS ID map type SHALL be the "Empty map" as defined in [RFC4563].

IDRi contains the identity of the Initiator. This identity SHOULD be included in the granted Ticket Policy.

IDRkms contains the identity of the KMS. It SHOULD be included, but it MAY be left out when it can be expected that the KMS has a single identity.

The Ticket Policy payload (TP) contains the desired Ticket Policy. It includes for instance, the ticket's validity period, the number of requested keys, and the identities of authorized responders (see Section 6.10).

4.2.1.2. Components of the REQUEST_INIT_PSK Message

The IDRi payload SHOULD be included but MAY be left out when it can be expected that the KMS can identify the Initiator by other means.

The IDRpsk payload is used to indicate the pre-shared key used. It MAY be omitted if the KMS can find the pre-shared key by other means.

The last payload SHALL be a Verification payload (V) where the authentication key (auth_key) is derived from the pre-shared key shared by the Initiator and the KMS (see Section 5.1.2 for key derivation specification). The MAC SHALL cover the entire REQUEST_INIT_PSK message as well as the identities of the involved parties (see Section 5.5 for the exact definition).

4.2.1.3. Components of the REQUEST_INIT_PK Message

The identity IDRi and certificate CERTi SHOULD be included, but they MAY be left out when it can be expected that the KMS can obtain the certificate in some other manner. If a certificate chain is to be provided, each certificate in the chain SHOULD be included in a separate CERT payload. The Initiator’s certificate MUST come first. Each following certificate MUST directly certify the one preceding it.

PKE contains the encrypted envelope key: PKE = E(PKkms, env_key). It is encrypted using the KMS’s public key (PKkms). If the KMS possesses several public keys, the Initiator can indicate the key used in the CHASH payload.

SIGNi is a signature covering the entire REQUEST_INIT_PK message, using the Initiator’s signature key (see Section 5.5 for the exact definition).

4.2.1.4. Processing the REQUEST_INIT Message

If the KMS can verify the integrity of the received message and the message can be correctly parsed, the KMS MUST check the Initiator’s authorization. If the Initiator is authorized to receive the requested ticket, possibly with a modified Ticket Policy, the KMS MUST send a REQUEST_RESP message. Unexpected payloads in the REQUEST_INIT message SHOULD be ignored. Errors are handled as described in Section 5.4.

4.2.1.5. Components of the REQUEST_RESP Message

The version, PRF func and CSB ID, #CS, and CS ID map type fields in the HDR payload SHALL be identical to the corresponding fields in the REQUEST_INIT message. The V flag has no meaning in this context. It SHALL be set to ’0’ by the KMS and ignored by the Initiator.

If one of the NTP timestamp types is used, the KMS SHALL generate a fresh timestamp value (unlike [RFC3830]), which may be used for clock synchronization. If the COUNTER timestamp type (see Section 6.6 of [RFC3830]) is used, the timestamp value MAY be equal to the one in the REQUEST_INIT message.

The TICKET payload carries the granted Ticket Policy and the Ticket Data (see Section 6.10). As the KMS decides which Ticket Policy to use, this may not be the same Ticket Policy as the Initiator requested. The Ticket Type and the Ticket Data depend on the granted Ticket Policy.

The KEMAC payload SHALL use the NULL authentication algorithm, as a MAC is included in the V payload. Depending on the type of REQUEST_INIT message, either the pre-shared key or the envelope key SHALL be used to derive the encr_key (and salt_key). Depending on the encryption algorithm, the salting key may go into the IV (see [RFC3830]). If the TP payload in the REQUEST_INIT message does not contain a KEMAC, it is RECOMMENDED that the KMS's default KEMAC include a single TGK. The KEMAC SHALL include an MPK (MIKEY Protection Key), MPKi, used as a pre-shared key to protect the messages in the Ticket Transfer exchange. If key forking (see Section 5.1.1) is used (determined by the Ticket Policy) a second MPK, MPKr, SHALL be included in the KEMAC. Then, MPKi SHALL be used to protect the TRANSFER_INIT message and MPKr SHALL be used to verify the TRANSFER_RESP message. The KEMAC is hence constructed as follows:

```
        KEMAC = E(encr_key, MPKi || [MPKr] || {TEK|TGK|GTGK})
```

The last payload SHALL be a Verification payload (V). Depending on the type of REQUEST_INIT message, either the pre-shared key or the envelope key SHALL be used to derive the auth_key. The MAC SHALL cover the entire REQUEST_RESP message as well as the REQUEST_INIT message (see Section 5.5 for the exact definition).

4.2.1.6. Processing the REQUEST_RESP Message

If the Initiator can verify the integrity of the received message and the message can be correctly parsed, the ticket and the associated session information SHALL be stored. Unexpected payloads in the REQUEST_RESP message SHOULD be ignored. Errors are handled as described in Section 5.4.

Before using the received ticket, the Initiator MUST check that the granted Ticket Policy is acceptable. If not, the Initiator SHALL discard and MAY send a new REQUEST_INIT message suggesting a different Ticket Policy than before.

4.2.2. Ticket Transfer

This exchange is used to transfer a ticket as well as session information from the Initiator to a Responder. The exchange is modeled after the pre-shared key mode [RFC3830], but instead of a pre-shared key, an MPK encoded in the ticket is used. The session keys are also encoded in the TICKET payload, but in some use cases (see Section 8) they need to be sent in a separate KEMAC payload. The session information may be sent from the Initiator to the Responder (similar to [RFC3830]) or from the Responder to the Initiator (similar to [RFC4738]). As the motive for this exchange is to setup a shared secret key between Initiator and Responder, the Responder cannot check the authenticity of the message before the ticket is resolved (by KMS or Responder). A full round trip is required if Responder key confirmation and freshness guarantee are needed.

```
   Initiator                                Responder

   TRANSFER_INIT =                ---->
   HDR, T, RANDRi, [IDRi],
      [IDRr], {SP}, TICKET,        < - -    TRANSFER_RESP =
      [KEMAC], V                            HDR, T, [RANDRr],
                                               [IDRr], [RANDRkms],
                                               {SP}, [KEMAC], V
```

4.2.2.1. Components of the TRANSFER_INIT Message

The TRANSFER_INIT message MUST always include the Header (HDR), Timestamp (T), and RANDRi payloads.

In HDR, the CSB ID (Crypto Session Bundle ID) SHALL be assigned as in [RFC3830]. The value of the V flag SHALL agree with the F flag in the Ticket Policy and it SHALL be ignored by the Responder.

The IDRi and IDRr payloads SHOULD be included, but IDRi MAY be left out if the Responder can identify the Initiator by other means, and IDRr MAY be left out when it can be expected that the Responder has a single identity.

Multiple SP payloads MAY be used both to indicate supported security policies for a specific crypto session (similar to [RFC4738]) and to specify security policies for different crypto sessions (similar to [RFC3830]).

The ticket payload (see Section 6.10) contains the Ticket Policy (see Section 6.10), Ticket Data (the default ticket type is defined in Appendix A), and Initiator Data. The Ticket Policy contains information intended for all parties involved, whereas the Ticket Data is only intended for the party that resolves the ticket. The Ticket Type provided in the Ticket Data is indicated in the Ticket Policy. The Initiator Data authenticates the Initiator when key forking (I flag) is used.

The KEMAC payload is handled in the same way as if it were sent in a later CSB update (see Section 5.2), with the only difference that the encr_key is always derived from MPKi and therefore accessible by all responders authorized to resolve the ticket. Initiator-specified keys MAY be used if the Initiator has pre-encrypted content and specific TEKs (Traffic Encryption Keys) need to be used (see Section 8). If indicated by the Ticket Policy (L flag), a KEMAC payload SHALL NOT be included.

The last payload SHALL be a Verification payload (V) where the authentication key (auth_key) is derived from the MPKi (see Section 5.1.2 for key derivation specification). The MAC SHALL cover the entire TRANSFER_INIT message as well as the identities of the involved parties (see Section 5.5 for the exact definition).

4.2.2.2. Processing the TRANSFER_INIT Message

As the Initiator and Responder do not have any pre-shared keys, the Responder cannot check the authenticity of the message before the ticket is resolved. The Responder SHALL however check that both the Ticket Policy and the security policies are acceptable. If they are not, the Responder SHALL reject without contacting the KMS. This is an early reject mechanism to avoid unnecessary KMS signaling when the Responder can conclude from the information at hand that it will not accept the connection. After the ticket has been resolved, the parsing of the TRANSFER_INIT message continues. Unexpected payloads in the TRANSFER_INIT message SHOULD be ignored. Errors are handled as described in Section 5.4. If the F flag in the Ticket Policy is set, the Responder MUST send a TRANSFER_RESP message.

4.2.2.3. Components of the TRANSFER_RESP Message

The version, PRF func and CSB ID fields in the HDR payload SHALL be identical to the corresponding fields in the TRANSFER_INIT message. The V flag has no meaning in this context. It SHALL be set to '0' by the Responder and ignored by the Initiator. The Responder SHALL update the CS ID map info so that each crypto session has exactly one security policy indicated. The Responder MUST provide Session Data (at least for SRTP) and SPI for each crypto session for which the Initiator has not supplied Session Data and SPI. If needed, the Responder MAY update Session Data and SPI provided by the Initiator. If the Responder adds crypto sessions, the #CS SHALL be updated.

If one of the NTP timestamp types is used, the Responder SHALL generate a fresh timestamp value (unlike [RFC3830]). If the COUNTER timestamp type (see Section 6.6 of [RFC3830]) is used, the timestamp value MAY be equal to the one in the TRANSFER_INIT message.

If indicated by the Ticket Policy (G flag), the Responder SHALL generate a fresh (pseudo-)random byte string RANDRr. RANDRr is used to produce Responder freshness guarantee in key derivations.

If the Responder receives an IDRr payload in the RESOLVE_RESP message, the same identity MUST be sent in an IDRr payload in the TRANSFER_RESP message. The identity sent in the IDRr payload in the

TRANSFER_RESP message (e.g., user1@example.com) MAY differ from the one sent in the IDRr payload in the TRANSFER_INIT message (e.g., IT-support@example.com).

If the Responder receives a RANDRkms payload in the RESOLVE_RESP message, the same RAND MUST be sent in a RANDRkms payload in the TRANSFER_RESP message.

The Responder MAY provide additional Security Policy payloads. The Responder SHOULD NOT resend SP payloads, which the Initiator supplied.

The KEMAC payload SHALL be handled exactly as if it was sent in a later CSB update, see Section 5.2. Responder-specified keys MAY be used if Responder has pre-encrypted content and specific TEKs (Traffic Encryption Keys) need to be used (see Section 8). If indicated by the Ticket Policy (M flag), a KEMAC payload SHALL NOT be included.

The last payload SHALL be a Verification payload (V) where the authentication key (auth_key) is derived from MPKi or MPKr' (depending on if key forking is used). The MAC SHALL cover the entire TRANSFER_RESP message as well as the TRANSFER_INIT message (see Section 5.5 for the exact definition).

4.2.2.4. Processing the TRANSFER_RESP Message

If the Initiator can verify the integrity of the received message and the message can be correctly parsed, the Initiator MUST check that any Responder-generated security policies are acceptable. If not, the Initiator SHALL discard and MAY send a new TRANSFER_INIT message to indicate supported security policies. Unexpected payloads in the TRANSFER_RESP message SHOULD be ignored. Errors are handled as described in Section 5.4.

4.2.3. Ticket Resolve

This exchange is used by the Responder to request that the KMS return the keys encoded in a ticket. The KMS does not need to be the same KMS that originally issued the ticket, see Section 10. A full round trip is required for the Responder to receive the keys. The Ticket Resolve exchange is OPTIONAL (depending on the Ticket Policy), and SHOULD only be used when the Responder is unable to resolve the ticket without assistance from the KMS. The initial message RESOLVE_INIT comes in two variants (independent from the used REQUEST_INIT variant). The first variant corresponds to the pre-shared key (PSK) method of [RFC3830].

```
   Responder                               KMS

   RESOLVE_INIT_PSK =             ---->
   HDR, T, RANDRr, [IDRr],
      [IDRkms], TICKET,            <----  RESOLVE_RESP
      [IDRpsk], V                         HDR, T, [IDRkms], KEMAC,
                                             [IDRr], [RANDRkms], V
```

The second variant corresponds to the public-key (PK) method of [RFC3830].

```
   Responder                               KMS

   RESOLVE_INIT_PK =              ---->
   HDR, T, RANDRr, [IDRr],
      {CERTr}, [IDRkms], TICKET,   <----  RESOLVE_RESP
      [CHASH], PKE, SIGNr                 HDR, T, [IDRkms], KEMAC,
                                             [IDRr], [RANDRkms], V
```

As the RESOLVE_INIT message MUST ensure the identity of the Responder to the KMS, it SHALL be protected by a MAC based on a pre-shared key or by a signature. The response message RESOLVE_RESP is the same for the two variants and SHALL be protected by using the pre-shared/envelope key indicated in the RESOLVE_INIT message.

Upon receiving the RESOLVE_INIT message, the KMS verifies that the Responder is authorized to resolve the ticket based on ticket and KMS policies. The KMS extracts the session information from the ticket and returns this to the Responder. Since the KMS resolved the ticket, the Responder is assured of the integrity of the Ticket Policy, which contains the identity of the peer that requested or created the ticket. If key forking is used (I flag), the Responder is also assured that the peer that requested or created the ticket also sent the TRANSFER_INIT message. The Responder can complete the session information it got from the Initiator with the additional session information received from the KMS.

4.2.3.1. Common Components of the RESOLVE_INIT Messages

The RESOLVE_INIT message MUST always include the Header (HDR), Timestamp (T), and RANDRr payloads.

The CSB ID (Crypto Session Bundle ID) SHALL be assigned as in [RFC3830]. The V flag MUST be set to '1' but SHALL be ignored by the KMS as a response is MANDATORY. As crypto sessions SHALL NOT be handled, the #CS MUST be set to '0' and the CS ID map type SHALL be the "Empty map" as defined in [RFC4563].

IDRkms SHOULD be included, but it MAY be left out when it can be expected that the KMS has a single identity.

The TICKET payload contains the Ticket Policy and Ticket Data that the Responder wants to have resolved.

4.2.3.2. Components of the RESOLVE_INIT_PSK Message

IDRr contains the identity of the Responder. IDRr SHOULD be included, but it MAY be left out when it can be expected that the KMS can identify the Responder in some other manner.

The IDRpsk payload is used to indicate the pre-shared key used. It MAY be omitted if the KMS can find the pre-shared key by other means.

The last payload SHALL be a Verification payload (V) where the authentication key (auth_key) is derived from the pre-shared key shared by the Responder and the KMS. The MAC SHALL cover the entire RESOLVE_INIT_PSK message as well as the identities of the involved parties (see Section 5.5 for the exact definition).

4.2.3.3. Components of the RESOLVE_INIT_PK Message

The identity IDRr and certificate CERTr SHOULD be included, but they MAY be left out when it can be expected that the KMS can obtain the certificate in some other manner. If a certificate chain is to be provided, each certificate in the chain SHOULD be included in a separate CERT payload. The Responder's certificate MUST come first. Each following certificate MUST directly certify the one preceding it.

PKE contains the encrypted envelope key: PKE = E(PKkms, env_key). It is encrypted using PKkms. If the KMS possesses several public keys, the Responder can indicate the key used in the CHASH payload.

SIGNr is a signature covering the entire RESOLVE_INIT_PK message, using the Responder's signature key (see Section 5.5 for the exact definition).

4.2.3.4. Processing the RESOLVE_INIT Message

If the KMS can verify the integrity of the received message, the message can be correctly parsed, and the Responder is authorized to resolve the ticket, the KMS MUST send a RESOLVE_RESP message. If key forking is used (I flag), the KMS SHALL also verify the integrity of the Initiator Data field in the TICKET payload. Unexpected payloads in the RESOLVE_INIT message SHOULD be ignored. Errors are handled as described in Section 5.4.

4.2.3.5. Components of the RESOLVE_RESP Message

The version, PRF func and CSB ID, #CS, and CS ID map type fields in the HDR payload SHALL be identical to the corresponding fields in the RESOLVE_INIT message. The V flag has no meaning in this context. It SHALL be set to '0' by the KMS and ignored by the Responder.

If one of the NTP timestamp types is used, the KMS SHALL generate a fresh timestamp value (unlike [RFC3830]), which may be used for clock synchronization. If the COUNTER timestamp type (see Section 6.6 of [RFC3830]) is used, the timestamp value MAY be equal to the one in the RESOLVE_INIT message.

The KEMAC payload SHALL use the NULL authentication algorithm, as a MAC is included in the V payload. Depending on the type of RESOLVE_INIT message, either the pre-shared key or the envelope key SHALL be used to derive the encr_key (and salt_key). Depending on the encryption algorithm, the salting key may go into the IV (see [RFC3830]). The KEMAC SHALL include an MPK (MPKi), used as a pre-shared key to protect the messages in the Ticket Transfer exchange. The KEMAC is hence constructed as follows:

```
        KEMAC = E(encr_key, MPKi || [MPKr'] || {TEK|TGK|GTGK})
```

If key forking (see Section 5.1.1) is used (determined by the I flag in the Ticket Policy), a second MPK (MPKr') SHALL be included in the KEMAC. Then, MPKi SHALL be used to verify the TRANSFER_INIT message and MPKr' SHALL be used to protect the TRANSFER_RESP message. The KMS SHALL also fork the MPKr and the TGKs. The modifier used to derive the forked keys SHALL be included in the IDRr and RANDRkms payloads, where IDRr is the identity of the endpoint that answered and RANDRkms is a fresh (pseudo-)random byte string generated by the KMS. The reason that the KMS MAY adjust the Responder's identity is so that it matches an identity encoded in the ticket.

The last payload SHALL be a Verification payload (V). Depending on the type of RESOLVE_INIT message, either the pre-shared key or the envelope key SHALL be used to derive the auth_key. The MAC SHALL cover the entire RESOLVE_RESP message as well as the RESOLVE_INIT message (see Section 5.5 for the exact definition).

4.2.3.6. Processing the RESOLVE_RESP Message

If the Responder can verify the integrity of the received message and the message can be correctly parsed, the Responder MUST verify the TRANSFER_INIT message with the MPKi received from the KMS. If key forking is used, the Responder SHALL also verify that the MAC field in the V payload in the TRANSFER_INIT message is identical to the MAC field in the Vi payload in the Initiator Data field in the TICKET payload. Unexpected payloads in the RESOLVE_RESP message SHOULD be ignored. Errors are handled as described in Section 5.4.

5. Key Management Functions

5.1. Key Derivation

For all messages in the Ticket Request and Ticket Resolve exchanges, the keys used to protect the MIKEY messages are derived from a pre-shared key or an envelope key. As crypto sessions SHALL NOT be handled, further keying material (i.e., TEKs) does not have to be derived.

In the Ticket Transfer exchange, the keys used to protect the MIKEY messages are derived from an MPK. If key forking is used, the KMS and the Initiator SHALL fork the MPKr and the TGKs (encoded in the ticket) based on a modifier, and different MPKs (MPKi and MPKr’) SHALL be used to protect the TRANSFER_INIT and TRANSFER_RESP messages. In addition, the Responder MAY generate a RAND used to give Responder key freshness guarantee.

The key hierarchy and its dependencies on TRANSFER_INIT message contents for the case without key forking and RANDRr are illustrated in Figure 4. The KEMAC shown is the KEMAC sent from the KMS to the Initiator and the Responder. The illustrated key derivations are done by the Initiator and the Responder.

```
                                +------+------------------+-----+------+
   KEMAC                        | MPKi |..................| TGK | SALT |
                                +--+---+------------------+--+--+--+---+
                                   | MPKi                    |     |
                                   v                         |     |
                       CSB ID    -----   auth_key    ------  |     |
                    +---------->| PRF |------------>| AUTH | |     |
                    |            -----               ------  |     |
                    |              ^                MAC |    |     |
                    |              | RAND               v    |     |
                 +--+--+------+----+---+--+--------+--+---+  |     |
   TRANSFER_INIT | HDR |......| RANDRi |..| TICKET |..| V |  |     |
                 +--+--+------+----+---+--+--------+--+---+  |     |
                    |              | RAND                    |     |
                    |              v                         |     |
                    |   CS ID    -----           TGK         |     |
                    +---------->| PRF |<---------------------+     |
                                 -----                             |
                                   | TEK                      SALT |
                                   v                               v
                                ---------------------------------------
                               |      Security Protocol, e.g., SRTP    |
                                ---------------------------------------
```

          Figure 4: Key hierarchy without key forking and RANDRr

The key hierarchy and its dependencies on TRANSFER_RESP message contents for the case with key forking and RANDRr are illustrated in Figure 5. The KEMAC shown is the KEMAC sent from the KMS to the Initiator. MOD is the modifier (IDRr, RANDRkms). The two key derivations that produce forked keys are done by the Initiator and the KMS, and the remaining two key derivations are done by the Initiator and the Responder. The random value RANDRi from the TRANSFER_INIT message is used as input to the derivation of the auth_key and may be used as input to the derivation of the TEK, but this is omitted from the figure. The protection of the TRANSFER_INIT message is done as in Figure 4.

```
                        +------+--------------------------+-----+------+
KEMAC                   | MPKr |..........................| TGK | SALT |
                        +--+---+--------------------------+--+--+--+---+
                           | MPKr                            |     |
                           v                                 |     |
                         -----   MPKr'                       |     |
                        | PRF |-------+                  TGK |     |
                         -----        |                      |     |
                           ^          v                      |     |
                   CSB ID  |        -----  auth_key  ------  |     |
                 +---------)------>| PRF |--------->| AUTH | |     |
                 |         |        -----            ------  |     |
                 |         | ID Data  ^             MAC |    |     |
                 |         | RAND     | RAND            v    |     |
              +--+--+---+--+--+---+---+----+----------+---+  |     |
TRANSFER_RESP | HDR |...| MOD |...| RANDRr |..........| V |  |     |
              +--+--+---+--+--+---+---+----+----------+---+  |     |
                 |         |          | RAND                 v     |
                 |         |          |          ID Data   -----   |
                 |         +----------)------------------>| PRF |  |
                 |                    |            RAND    -----   |
                 |                    v                      |     |
                 |       CS ID      -----         TGK'       |     |
                 +--------------->| PRF |<------------------+     |
                                    -----                          |
                                      | TEK                   SALT |
                                      v                            v
                                ---------------------------------------
                               |      Security Protocol, e.g., SRTP    |
                                ---------------------------------------
```

Figure 5: Key hierarchy with key forking and RANDRr

The labels in the key derivations SHALL NOT include entire RANDR payloads, only the fields RAND length and RAND from the corresponding payload.

5.1.1. Deriving Forked Keys

When key forking is used (determined by the I flag in the Ticket Policy), the MPKr and TGKs (encoded in the ticket) SHALL be forked. The TEKs and GTGKs (Group TGKs), however, SHALL NOT be forked. This key forking is done by the KMS and the Initiator using the PRF (Pseudorandom Function) indicated in the Ticket Policy. The parameters for the PRF are:

```
inkey:     : MPKr or TGK
inkey_len  : bit length of the inkey
label      : constant || 0xFF || 0xFFFFFFFF || 0x00 ||
             length ID Data || ID Data || length RANDRkms || RANDRkms
outkey_len : desired bit length of the outkey (MPKr', TGK')
             SHALL be equal to inkey_len
```

where the ID Data field is taken from the IDRr payload sent in the RESOLVE_RESP and TRANSFER_RESP messages. Length ID Data is the length of the ID Data field in bytes as a 16-bit unsigned integer. Length RANDRkms is the length of RANDRkms in bytes as an 8-bit unsigned integer. The constant depends on the derived key type as summarized below.

Derived key	Constant
MPKr'	0x2B288856
TGK'	0x1512B54A

Table 5.1: Constants for forking key derivation

The constants are taken from the decimal digits of e as described in [RFC3830].

5.1.2. Deriving Keys from an Envelope Key/PSK/MPK

This derivation is used to form the keys used to protect the MIKEY messages. For the Ticket Request and Ticket Resolve exchanges, the keys used to protect the MIKEY messages are derived from a pre-shared key or an envelope key. For the Ticket Transfer exchange, the keys are derived from an MPK. If key forking is used, different MPKs (MPKi and MPKr') SHALL be used to protect the TRANSFER_INIT and TRANSFER_RESP messages. The initial messages SHALL be protected with keys derived using the following parameters:

```
inkey:     : pre-shared key, envelope key, or MPKi
inkey_len  : bit length of the inkey
label      : constant || 0xFF || CSB ID || 0x01 ||
             length RANDRi || [RANDRi] || length RANDRr || [RANDRr]
outkey_len : desired bit length of the outkey (encr_key,
             auth_key, salt_key)
```

The response messages SHALL be protected with keys derived using the following parameters:

```
inkey:      : pre-shared key, envelope key, MPKi, or MPKr'
inkey_len   : bit length of the inkey
label       : constant || 0xFF || CSB ID || 0x02 ||
              length RANDRi || [RANDRi] || length RANDRr || [RANDRr]
outkey_len : desired bit length of the outkey (encr_key,
              auth_key, salt_key)
```

The constant depends on the derived key type as defined in Section 4.1.4 of [RFC3830]. The 32-bit CSB ID field is taken from the HDR payload. RANDRi SHALL be included in the derivation of keys used to protect the Ticket Request and Ticket Transfer exchanges. RANDRr SHALL be included in the derivation of keys used to protect the Ticket Resolve exchange and in the derivation of keys used to protect TRANSFER_RESP if the Ticket Policy determines that it shall be present in the TRANSFER_RESP message (G flag). Length RANDRi is the length of RANDRi in bytes as an 8-bit unsigned integer, and Length RANDRr is the length of RANDRr in bytes as an 8-bit unsigned integer. If RANDRi is omitted, length RANDRi SHALL be 0 and if RANDRr is omitted, length RANDRr SHALL be 0. Note that at least one of RANDRi and RANDRr is always used.

5.1.3. Deriving Keys from a TGK/GTGK

This only affects the Ticket Transfer exchange. In the following, we describe how keying material is derived from a TGK/GTGK. If key forking is used, any TGK encoded in the ticket SHALL be forked, and the forked key TGK' SHALL be used. The key derivation method SHALL be executed using the PRF indicated in the HDR payload. The parameters for the PRF are:

```
inkey:      : TGK, TGK', or GTGK
inkey_len   : bit length of the inkey
label       : constant || CS ID || 0xFFFFFFFF || 0x03 ||
              length RANDRi || [RANDRi] || length RANDRr || [RANDRr]
outkey_len : desired bit length of the outkey (TEK, encr_key,
              auth_key, salt_key)
```

The constant depends on the derived key type as defined in Section 4.1.3 of [RFC3830]. If a salting key is present in the key data sub-payload, a security protocol in need of a salting key SHALL use this salting key and a new salting key SHALL NOT be derived. The 8-bit CS ID field is given by the CS ID map info field in the HDR payload. RANDRi SHALL be included if the Ticket Policy determines that it shall be used (H flag). RANDRr SHALL be included if the Ticket Policy determines that it shall be present in the TRANSFER_RESP message (G flag). Length RANDRi is the length of RANDRi in bytes as an 8-bit unsigned integer, and Length RANDRr is the length of RANDRr in bytes as an 8-bit unsigned integer. If RANDRi or RANDRr is omitted the corresponding length SHALL be 0. Note that at least one of RANDRi and RANDRr MUST be used.

5.2. CSB Updating

Similar to [RFC3830], MIKEY-TICKET provides a means of updating the CSB (Crypto Session Bundle), e.g., transporting a new TEK/TGK/GTGK or adding new crypto sessions. The CSB updating is done by executing the Ticket Transfer exchange again, e.g., before a TEK expires or when a new crypto session is needed. The CSB updating can be started by the Initiator:

```
Initiator                               Responder

TRANSFER_INIT =                   ---->
HDR, T, [IDRi], [IDRr],
   {SP}, [KEMAC], V                < - -  TRANSFER_RESP =
                                          HDR, T, [IDRr],
                                          {SP}, [KEMAC], V
```

The CSB updating can also be started by the Responder:

```
Responder                               Initiator

TRANSFER_INIT =                   ---->
HDR, T, [IDRr], [IDRi],
   {SP}, [KEMAC], V                < - -  TRANSFER_RESP =
                                          HDR, T, [IDRi],
                                          {SP}, [KEMAC], V
```

The new message exchange MUST use the same CSB ID as the initial exchange but MUST use new timestamps. The crypto sessions negotiation (#CS field, CS ID map info field, and SP payloads) are handled as in the initial exchange. In the TRANSFER_INIT message the V flag SHALL be used to indicate whether or not a response message is expected. Static payloads such as RANDRi, RANDRr, RANDRkms, and TICKET that were provided in the initial exchange SHOULD NOT be included unless they are needed by a specific use case. New RANDS or TICKETs MUST NOT be included. The reason that new RANDS SHALL NOT be used is that if several TGKs are used, the peers would need to keep track of which RANDS to use for each TGK. This adds unnecessary complexity. Both messages SHALL be protected with the same keys (derived from MPKi or MPKr’) that protected the last message (TRANSFER_INIT or TRANSFER_RESP) in the initial exchange.

New keying material MAY be sent in a KEMAC payload. If indicated by the Ticket Policy (L and M flags), KEMAC payloads SHALL NOT be included. In the TRANSFER_RESP message, a session key MUST be provided for each crypto session. The KEMAC SHALL use the NULL authentication algorithm, as a MAC is included in the V payload. The encr_key (and salt_key) SHALL be derived from the MPK (MPKi or MPKr'). Depending on the encryption algorithm, the salting key may go into the IV (see [RFC3830]). If a new TGK is exchanged, it SHALL NOT be forked. The KEMAC is hence constructed as follows:

```
KEMAC = E(encr_key, (TEK|TGK|GTGK))
```

5.3. Ticket Reuse

MIKEY-TICKET includes features aiming to offload the KMS from receiving ticket requests. One such feature is that tickets may be reused. This means that a user may request a ticket for media sessions with another user and then under the ticket's validity period use this ticket to protect several media sessions with that user.

When reusing a ticket that has been used in a previous Ticket Transfer exchange, a new Ticket Transfer exchange is executed. The new exchange MUST use a new CSB ID, a new timestamp, and new RANDs (RANDRi, RANDRr). If the Responder has resolved the ticket before, the Responder does not need to resolve the ticket again. In that case, the same modifier (IDRr, RANDRkms) SHALL be used. If the Ticket Policy forbids reuse (J flag), the ticket MUST NOT be reused. Note that such reuse cannot be detected by a stateless KMS. When group keys are used, ticket reuse leaves the Initiator responsible to ensure that group membership has not changed since the ticket was last used. (Otherwise, unauthorized responders may gain access to the group communication.) Thus, if group dynamics are difficult to verify, the Initiator SHOULD NOT initiate ticket reuse.

When key forking is used, only the user that requested the ticket has access to the encoded master keys (MPKr, TGKs). Because of this, no one else can initiate a Ticket Transfer exchange using the ticket.

5.4. Error Handling

If a fatal error occurs during the parsing of a message, the message SHOULD be discarded, and an Error message SHOULD be sent to the other party (Initiator, Responder, KMS). If a failure is due to the inability to authenticate the peer, the message SHALL be discarded, the Error message is OPTIONAL, and the caveats in Section 5.1.2 of [RFC3830] apply. Error messages may be used to report errors in both initial and response messages, but not in Error messages.

In the Ticket Request and Ticket Resolve exchanges, the Error message MAY be authenticated with a MAC or a signature. The Error message is hence constructed as follows:

```
                Error message = HDR, T, (ERR), [V|SIGNx]
```

where x is in the set {i, r, kms} (Initiator, Responder, KMS). Unexpected payloads in the Error message SHOULD be ignored.

In the Ticket Transfer exchange, the Error message MAY be authenticated with a MAC. If the suggested security policies are not supported, the Error message SHOULD include the supported parameters. The Error message is hence constructed as follows:

```
                Error message = HDR, T, (ERR), {SP}, [V]
```

In Error messages, the version, PRF func, and CSB ID fields in the HDR payload SHALL be identical to the corresponding fields in the message where the error occurred. The V field SHALL be set to '0' and be ignored.

If one of the NTP timestamp types is used, a fresh timestamp value SHALL be used. If the COUNTER timestamp type (see Section 6.6 of [RFC3830]) is used, the timestamp value MAY be equal to the one in the message where the error occurred.

The MAC/Signature in the V/SIGN payloads covers the entire Error message, except the MAC/Signature field itself. The auth_key SHALL be the same as in the message where the error occurred.

5.5. MAC/Signature Coverage

The MAC/Signature in the V/SIGN payloads covers the entire MIKEY message, except the MAC/Signature field itself. For initial messages, the identities (not whole payloads) of the parties involved MUST directly follow the MIKEY message in the Verification MAC/Signature calculation. In the TRANSFER_INIT message, the MAC SHALL NOT cover the Initiator Data length and Initiator Data fields in the TICKET payload. Note that in the Transfer Exchange, Identity_r in TRANSFER_RESP (e.g., user1@example.com) MAY differ from that appearing in TRANSFER_INIT (e.g., IT-support@example.com). For response messages, the entire initial message (including the MAC/Signature field) MUST directly follow the MIKEY message in the Verification MAC/Signature calculation (the identities are implicitly covered as they are covered by the initial message's MAC/Signature).

Message type	MAC/Signature coverage
REQUEST_INIT	REQUEST_INIT || Identity_i || Identity_kms
REQUEST_RESP	REQUEST_RESP || REQUEST_INIT
TRANSFER_INIT	TRANSFER_INIT || Identity_i || Identity_r
TRANSFER_RESP	TRANSFER_RESP || TRANSFER_INIT
RESOLVE_INIT	RESOLVE_INIT || Identity_r || Identity_kms
RESOLVE_RESP	RESOLVE_RESP || RESOLVE_INIT
Error message	Error message

Table 5.2: MAC/Signature coverage

6. Payload Encoding

This section does not describe all the payloads that are used in the new message types. It describes in detail the new TR, IDR, RANDR, TP, and TICKET payloads. For the other payloads, only the additions and changes compared to [RFC3830] are described. For a detailed description of the other MIKEY payloads, see [RFC3830]. Note that the fields with variable length are byte aligned and not 32-bit aligned.

6.1. Common Header Payload (HDR)

For the Common Header Payload, new values are added to the Data Type, Next Payload, PRF func, and CS ID map type name spaces.

* Data Type (8 bits): describes the type of message.

Data Type	Value	Comment
REQUEST_INIT_PSK	11	Ticket request initial message (PSK)
REQUEST_INIT_PK	12	Ticket request initial message (PK)
REQUEST_RESP	13	Ticket request response message
TRANSFER_INIT	14	Ticket transfer initial message
TRANSFER_RESP	15	Ticket transfer response message
RESOLVE_INIT_PSK	16	Ticket resolve initial message (PSK)
RESOLVE_INIT_PK	17	Ticket resolve initial message (PK)
RESOLVE_RESP	18	Ticket resolve response message

Table 6.1: Data Type (Additions)

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

Next Payload	Value	Section
TR	13	6.4
IDR	14	6.6
RANDR	15	6.8
TP	16	6.10
TICKET	17	6.10

Table 6.2: Next Payload (Additions)

* V (1 bit): flag to indicate whether a response message is expected ('1') or not ('0'). It MUST be set to '0' and ignored in all messages except TRANSFER_INIT messages used for CSB updating (see Section 5.2).

* PRF func (7 bits): indicates the PRF function that has been/will be used for key derivation. Besides the PRFs already defined in [RFC3830] the following additional PRF may be used.

PRF func	Value
PRF-HMAC-SHA-256	1

Table 6.3: PRF func (Additions)

The new PRF SHALL be constructed as described in Section 4.1.2 of [RFC3830] with the differences that HMAC-SHA-256 (see Section 6.2) SHALL be used instead of HMAC-SHA-1 and the value 256 SHALL be used instead of 160. This corresponds to the full output length of SHA-256.

* #CS (8 bits): indicates the number of crypto sessions in the CS ID map info.

* CS ID map type (8 bits): specifies the method of uniquely mapping crypto sessions to the security protocol sessions. In the Ticket Transfer exchange the new GENERIC-ID map type, which is intended to eliminate the limitations with the existing SRTP-ID map type, SHOULD be used. The map type SRTP-ID SHALL NOT be used.

CS ID map type	Value
GENERIC-ID	2

Table 6.4: CS ID map type (Additions)

   * CS ID map info (variable length): identifies and maps the crypto
      sessions to the security protocol sessions for which security
      associations should be created.

6.1.1. The GENERIC-ID Map Type

For the GENERIC-ID map type, the CS ID map info consists of #CS number of blocks, each mapping policies, session data (e.g., SSRC), and key to a specific crypto session.

```
    0                   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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !     CS ID     !   Prot type   !S!      #P     ! Ps (OPTIONAL) ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !      Session Data Length      !    Session Data (OPTIONAL)    ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  SPI Length   !                SPI (OPTIONAL)                 ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * CS ID (8 bits): defines the CS ID to be used for the crypto
      session.

   * Prot type (8 bits): defines the security protocol to be used for
      the crypto session. Allowed values are the ones defined for the
      Prot type field in the SP payload (see Section 6.10 of [RFC3830]).

   * S (1 bit): flag that MAY be used by the Session Data.

   * #P (7 bits): indicates the number of security policies provided
      for the crypto session. In response messages, #P SHALL always be
      exactly 1. So if #P = 0 in an initial message, a security profile
      MUST be provided in the response message. If #P > 0, one of the
      suggested policies SHOULD be chosen in the response message. If
      needed (e.g., in group communication, see Section 9), the
      suggested policies MAY be changed.

   * Ps (variable length): lists the policies for the crypto session.
      It SHALL contain exactly #P policies, each having the specified
      Prot type.

```
       0                   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  !  Policy_no_2  !      ...      ! Policy_no_#P  !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

      * Policy_no_i (8 bits): a policy_no that corresponds to the
         policy_no of a SP payload. In response messages, the policy_no
         may refer to a SP payload in the initial message.

   * Session Data Length (16 bits): the length of Session Data (in
      bytes). For the Prot type SRTP, Session Data MAY be omitted in
      the initial message (length = 0), but it MUST be provided in the
      response message.

   * Session Data (variable length): contains session data for the
      crypto session. The type of Session Data depends on the specified
      Prot type. The Session Data for the Prot type SRTP is defined
      below. The S flag is used to indicate whether the ROC and SEQ
      fields are provided ('1') or if they are omitted ('0').

```
       0                   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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      !                              SSRC                             !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      !                        ROC (OPTIONAL)                         !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      !         SEQ (OPTIONAL)         !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

      * SSRC (32 bits): specifies the SSRC that MUST be used for the
         crypto session. Note that unlike [RFC3830], an SSRC field set
         to '0' has no special meaning.

      * ROC (32 bits): current/initial rollover counter. If the
         session has not started, this field is set to '0'.

      * SEQ (16 bits): current/initial sequence number.

   * SPI Length (8 bits): the length of SPI (in bytes). SPI MAY be
      omitted in the initial message (length = 0), but it MUST be
      provided in the response message.

   * SPI (variable length): the SPI (or MKI) corresponding to the
      session key to (initially) be used for the crypto session. This
      does not exclude other keys to be used. All keys MUST belong to
      the crypto session bundle.

6.2. Key Data Transport Payload (KEMAC)

   For the KEMAC payload, new encryption and authentication algorithms
   are defined.

* Encr alg (8 bits): the encryption algorithm used to encrypt the Encr data field. Besides the algorithms already defined in [RFC3830], the following additional encryption algorithm may be used.

Encr alg	Value	Comment
AES-CM-256	3	AES-CM using a 256-bit key

Table 6.5: Encr alg (Additions)

The new encryption algorithm is defined as described in Section 4.2.3 of [RFC3830] with the only difference being that a 256-bit key SHALL be used.

* MAC alg (8 bits): specifies the authentication algorithm used. Besides the algorithms already defined in [RFC3830], the following additional authentication algorithm may be used.

MAC alg	Value	Length
HMAC-SHA-256-256	2	256 bits

Table 6.6: MAC alg (Additions)

The new authentication algorithm is Hash-based Message Authentication Code (HMAC) [RFC2104] in conjunction with SHA-256 [FIPS.180-3]. It SHALL be used with a 256-bit authentication key.

6.2.1. Key Data Sub-Payload

For the key data sub-payload, new types of keys are defined. The Group TGK (GTGK) is used as a regular TGK, with the difference that it SHALL NOT be forked. It is intended to enable the establishment of a group TGK when key forking is used. The MIKEY Protection Key (MPK) is used to protect the MIKEY messages in the Ticket Transfer exchange. The MPK is used as the pre-shared key in the pre-shared key method of [RFC3830]; however, it is not known by the Responder before the ticket has been resolved.

An SPI (or MKI) MUST be specified for each key (see Section 6.13 of [RFC3830]).

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

Type	Value	Comments
GTGK	4	Group TGK
GTGK+SALT	5	Group TGK + SALT
MPK	6	MIKEY Protection Key

Table 6.7: Key Data Type (Additions)

6.3. Timestamp Payload (T)

For the timestamp payload, a new type of timestamp is defined. The new type is intended to be used when defining validity periods, where fractions of seconds seldom matter. The NTP-UTC-32 string contains four bytes, in the same format as the first four bytes in the NTP timestamp format, defined in [RFC4330]. This represents the number of seconds since 0h on 1 January 1900 with respect to the Coordinated Universal Time (UTC). On 7 February 2036, the time value will overflow. [RFC4330] describes a procedure to extend the time to 2104 and this procedure is MANDATORY to support.

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

TS Type	Value	Length
NTP-UTC-32	3	32 bits

Table 6.8: TS Type (Additions)

NTP-UTC-32 SHALL be padded to a 64-bit NTP-UTC timestamp (with zeroes in the fractional second part) when a 64-bit timestamp is required (e.g. IV creation in AES-CM-128 and AES-CM-256).

6.4. Timestamp Payload with Role Indicator (TR)

The TR payload uses all the fields from the standard timestamp payload (T) but expands it with a new field describing the role of the timestamp. Whereas the TS Type describes the type of the TS Value, the TS Role describes the meaning of the timestamp itself. The TR payload is intended to eliminate ambiguity when a MIKEY message contains several timestamp payloads (e.g., in the Ticket Policy).

```
    0                   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 Role    !    TS Type    !    TS Value   ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* TS Role (8 bits): specifies the sort of timestamp.

TS Role	Value
Time of issue (TRi)	1
Start of validity period (TRs)	2
End of validity period (TRe)	3
Rekeying interval (TRr)	4

Table 6.9: TS Role

6.5. ID Payload (ID)

For the ID payload, a new ID Type byte string is defined. The byte string type is intended to be used when the ID payload is used to identify a pre-shared key. Contrary to the previously defined ID Types (URI, Network Access Identifier), the byte string does not have any encoding rules.

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

ID Type	Value
Byte string	2

Table 6.10: ID Type (Additions)

6.6. ID Payload with Role Indicator (IDR)

The IDR payload uses all the fields from the standard identity payload (ID) but expands it with a new field describing the role of the ID payload. Whereas the ID Type describes the type of the ID Data, the ID Role describes the meaning of the identity itself. The IDR payload is intended to eliminate ambiguity when a MIKEY message contains several identity payloads. The IDR payload MUST be used instead of the ID payload in all MIKEY-TICKET messages.

```
    0                   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 Role    !    ID Type    !     ID len
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ID len (cont) !                   ID Data                     ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* ID Role (8 bits): specifies the sort of identity.

ID Role	Value
Initiator (IDRi)	1
Responder (IDRr)	2
KMS (IDRkms)	3
Pre-Shared Key (IDRpsk)	4
Application (IDRapp)	5

Table 6.11: ID Role

IDRapp is intended to specify the authorized Application IDs (see Sections 5.1.3 and 6.10)

6.7. Cert Hash Payload (CHASH)

* Hash func (8 bits): indicates the hash function that is used. Besides the hash functions already defined in [RFC3830], the following hash function may be used.

Hash func	Value	Hash Length
SHA-256	2	256 bits

Table 6.12: Hash func (Additions)

The SHA-256 hash function is defined in [FIPS.180-3].

6.8. RAND Payload with Role Indicator (RANDR)

The RANDR payload uses all the fields from the standard RAND payload (RAND) but expands it with a new field describing the role (the generating entity) of the RAND. The RANDR payload is intended to eliminate ambiguity when a MIKEY message contains several RAND payloads.

```
 0                   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 Role   !  RAND length  !     RAND      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* RAND Role (8 bits): specifies the entity that generated the RAND.

RAND Role	Value
Initiator (RANDRi)	1
Responder (RANDRr)	2
KMS (RANDRkms)	3

Table 6.13: RAND Role

6.9. Error Payload (ERR)

For the key data sub-payload, new types of errors are defined.

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

Error no	Value	Comments
Invalid TICKET	14	Ticket Type not supported
Invalid TPpar	15	TP parameters not supported

Table 6.14: Error no (Additions)

6.10. Ticket Policy Payload (TP) / Ticket Payload (TICKET)

Note that the Ticket Policy payload (TP) and the Ticket Payload (TICKET) are two different payloads (having different payload identifiers). However, as they share much of the payload structure, they are described in the same section.

The Ticket Policy payload contains a desired Ticket Policy and does not include the Ticket Data length, Ticket Data, Initiator Data length, or Initiator Data fields. The ticket payload contains the granted Ticket Policy as well as Ticket Data (the default ticket type is defined in Appendix A). The Ticket Policy contains information intended for all parties involved whereas the Ticket Data is only intended for the party that resolves the ticket. The Ticket Type provided in the Ticket Data is indicated in the Ticket Policy. When key forking is used (I flag), the Initiator Data authenticates the Initiator.

Note that the flags are not independent: NOT D implies L, G implies F, NOT G implies H, NOT H implies G, I implies E, K implies D, and M implies F. The F flag SHALL be set to '1' when the I flag (key forking) is set to '1' and a TGK is encoded in the ticket.

```
    0                   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  !          Ticket Type          !    Subtype    !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !    Version    !   PRF Func  !D!E!F!G!H!I!J!K!L!M!N!O!   Res   !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !        TP Data length         !            TP Data            ~
   +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
   !      Ticket Data length       !          Ticket Data          ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !     Initiator Data length     !   Initiator Data (OPTIONAL)   ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

* Ticket Type (16 bits): specifies the Ticket Type used.

Ticket Type	Value	Comments
MIKEY Base Ticket	1	Defined in Appendix A
3GPP Base Ticket	2	Used and specified by 3GPP

Table 6.15: Ticket Type

Subtype = 0x01 and Version = 0x01 refers to MIKEY Base Ticket as defined in this document.

* Subtype (8 bits): specifies the ticket subtype used.

* Version (8 bits): specifies the ticket subtype version used.

* PRF Func (7 bits): specifies the PRF that SHALL be used for key forking.

* D (1 bit): flag to indicate whether the ticket was generated by the KMS ('1') or by the Initiator ('0').

* E (1 bit): flag to indicate whether the Ticket Resolve exchange is MANDATORY ('1') or if the Responder MAY resolve the ticket ('0').

* F (1 bit): flag to indicate whether the TRANSFER_RESP message SHALL be sent ('1') or if it SHALL NOT be sent ('0').

* G (1 bit): flag to indicate whether the Responder SHALL generate RANDRr ('1') or if the Responder SHALL NOT generate RANDRr ('0').

* H (1 bit): flag to indicate whether RANDRi SHALL be used when deriving keys from a TGK/GTGK ('1') or if RANDRi SHALL NOT be used ('0').

* I (1 bit): flag to indicate whether key forking SHALL be used ('1') or if key forking SHALL NOT be used ('0').

* J (1 bit): flag to indicate whether the ticket MAY be reused ('1') and therefore MAY be cached or if it SHALL NOT be reused ('0').

* K (1 bit): flag to indicate whether the KMS changed the desired Ticket Policy or the desired KEMAC ('1') or if it did not ('0'). In the TP payload, it SHALL be set to '0' by the Initiator and ignored by the KMS.

* L (1 bit): flag to indicate whether the Initiator MAY supply session keys ('1') or if the Initiator SHALL NOT supply session keys ('0').

* M (1 bit): flag to indicate whether the Responder MAY supply session keys ('1') or if the Responder SHALL NOT supply session keys ('0').

* N (1 bit): flag to indicate whether an Initiator following this specification can initiate a TRANSFER_INIT message using the ticket ('1') or if additional processing is required ('0'). If the flag is set to '0', the Initiator SHOULD follow the processing in the specification of the received Ticket Type.

* O (1 bit): flag to indicate whether a Responder following this specification can process a TRANSFER_INIT message containing the ticket ('1') or if additional processing is required ('0'). If the flag is set to '0', the Responder SHOULD follow the processing in the specification of the received Ticket Type.

* Res (5 bits): reserved for future use.

* TP Data length (16 bits): length of TP Data (in bytes).

* TP Data (variable length): The first 8 bits identify the first payload. The rest of TP Data SHALL be constructed of MIKEY payloads. Unexpected payloads in the TP Data SHOULD be ignored.

```
       TP Data = First Payload, [IDRkms], [IDRi], [TRs],
                 [TRe], [TRr], [KEMAC], {IDRapp}, (IDRr)
```

IDRkms contains the identity of a KMS that can resolve the ticket.

IDRi contains the identity of the peer that requested or created the ticket.

TRs is the start of the validity period. TRs SHALL be interpreted as being in the range 1968-2104 as described in [RFC4330]. An omitted TRs means that the validity period has no defined beginning.

TRe is the end of the validity period. TRe SHALL be interpreted as being in the range 1968-2104 as described in [RFC4330]. An omitted TRe means that the validity period has no defined end.

TRr indicates how often rekeying MUST be done. TS Type SHALL be NTP-UTC-32 and the time between two rekeyings SHALL NOT be longer than the number of seconds in the integer part of the timestamp. How the rekeying is done is implementation specific.

The KEMAC payload may be used to indicate the number of requested keys and specify other key information (key type, key length, and KV (key validity) data). The KEMAC payload SHALL use the NULL encryption algorithm and the NULL authentication algorithm, as a MAC is included in the V payload. The KEMAC is hence constructed as follows:

```
KEMAC = {TEK|TGK|GTGK}
```

The Key Data fields SHALL be set to ’0’ by the Initiator and ignored by the KMS. The KEMAC SHALL NOT be present in the granted Ticket Policy.

IDRapp is an identifier for an authorized application ID. The application IDs are implementation specific. If no IDRapp payloads are supplied, all application IDs are authorized.

IDRr is the identity of a responder or a group of responders that are authorized to resolve the ticket. If there is more than one responder identity, each responder identity SHALL be included in a separate IDR payload.

* Ticket Data length (16 bits): the length of the Ticket Data field (in bytes). Not present in the TP payload.

* Ticket Data (variable length): contains the Ticket Data. Not present in the TP payload.

* Initiator Data length (16 bits): the length of the Initiator Data field (in bytes). Not present in the TP payload.

* Initiator Data (variable length): Not present in the TP payload. SHALL be inserted by the Initiator if and only if key forking is used (I flag). The first 8 bits identifies the first payload. The rest of Initiator Data SHALL be constructed of MIKEY payloads. Unexpected payloads in the Initiator Data SHOULD be ignored.

```
             Initiator Data = First Payload, Vi, Vr
```

  The Vi payload SHALL be identical to the V payload in the TRANSFER_INIT message.

  The last payload (Vr) SHALL be a Verification payload where the MAC SHALL cover the entire Initiator Data field except the MAC field itself. The authentication algorithm SHALL be the same as used for the Vi payload. The authentication key (auth_key) SHALL be derived from MPKr (not forked) using the following parameters:

```
inkey:      : MPKr
inkey_len   : bit length of the inkey
label       : constant || 0xFF || 0xFFFFFFFF || 0x04
outkey_len  : desired bit length of the outkey (encr_key,
              auth_key, salt_key)
```

  The constant depends on the derived key type as defined in Section 4.1.4 of [RFC3830].

7. Transport Protocols

MIKEY messages are not tied to any specific transport protocols. In [RFC4567], extensions for SDP and RTSP to carry MIKEY messages (and therefore MIKEY-TICKET messages) are defined. The messages in the Ticket Transfer exchange (TRANSFER_INIT, TRANSFER_RESP) are preferably included in the session setup signaling (e.g., SIP INVITE and 200 OK). However, it may not be suitable for the MIKEY-TICKET exchanges that do not establish keying material for media sessions (Ticket Request and Ticket Resolve) to be carried in SDP or RTSP. If SDP or RTSP is not used, the transport protocol needs to be defined. In [3GPP.33.328], it is defined how the Ticket Request and Ticket Resolve exchanges are carried over HTTP.

8. Pre-Encrypted Content

The default setting is that the KMS supplies the session keys (encoded in the ticket). This is not possible if the content is pre-encrypted (e.g., Video on Demand). In such use cases, the key exchange is typically reversed and MAY be carried out as follows. The Initiator sends a ticket without encoded session keys to the Responder in a TRANSFER_INIT message. The Responder has access to the TEKs used to protect the requested content, but may not be streaming the content. The Responder includes the TEK in the TRANSFER_RESP message, which is sent to the Initiator.

```
   +---+                                                           +---+
   | I |                                                           | R |
   +---+                                                           +---+
                               TRANSFER_INIT
     ---------------------------------------------------------------->
                               TRANSFER_RESP {KEMAC}
     <----------------------------------------------------------------
```

Figure 6: Distribution of pre-encrypted content

9. Group Communication

What has been discussed up to now can also be used for group communication. The MIKEY signaling for multi-party sessions can be centralized as illustrated in Figure 7.

```
   +---+                           +---+                           +---+
   | A |                           | B |                           | C |
   +---+                           +---+                           +---+
              Ticket Transfer
     <-------------------------------->       Ticket Transfer
     <---------------------------------------------------------------->
```

Figure 7: Centralized signaling around party A

or decentralized as illustrated in Figure 8.

```
   +---+                           +---+                           +---+
   | A |                           | B |                           | C |
   +---+                           +---+                           +---+
              Ticket Transfer
     <-------------------------------->       Ticket Transfer
                                     <--------------------------------->
```

Figure 8: Decentralized signaling

In the decentralized scenario, the identities of B and C SHALL be used in the second Ticket Transfer exchange. Independent of the how the MIKEY signaling is done, a group key may be used as session key.

If a group key is used, the group key and session information may be pushed to all group members (similar to [RFC3830]), or distributed when requested (similar to [RFC4738]). If a TGK/GTGK is used as a group key, the same RANDs MUST be used to derive the session keys in all Ticket Transfer exchanges. Also note caveats with ticket reuse in group communication settings as discussed in Section 5.3.

9.1. Key Forking

When key forking is used, only the user that requested the ticket can initiate a Ticket Transfer exchange using that ticket, see Section 5.3. So if a group key is to be distributed, the MIKEY signaling MUST be centralized to the party that initially requested the ticket, or different tickets needs to be used in each Ticket Transfer exchange and the group key needs to be sent in a KEMAC.

Another consideration is that different users get different session keys if TGKs (encoded in the ticket) are used.

10. Signaling between Different KMSs

A user can in general only be expected to have a trust relation with a single KMS. Different users might therefore use tickets issued by different KMSs using only locally known keys. Thus, if users with trust relations to different KMSs are to be able to establish a secure session with each other, the KMSs involved have to cooperate and there has to be a trust relation between them. The KMSs SHALL be mutually authenticated and signaling between them SHALL be integrity and confidentiality protected. The technical means for the inter-KMS security is however outside the scope of this specification. Under these assumptions, the following approach MAY be used.

```
+---+                 +---+              +-------+           +-------+
| I |                 | R |              | KMS R |           | KMS I |
+---+                 +---+              +-------+           +-------+
      TRANSFER_INIT
  -------------------->    RESOLVE_INIT
                    - - - - - - - - - - ->    RESOLVE_INIT
                                         - - - - - - - - - - ->
                                              RESOLVE_RESP
                           RESOLVE_RESP    <- - - - - - - - - - -
      TRANSFER_RESP   < - - - - - - -  - - -
  <--------------------
```

Figure 9: Routing of resolve messages

If the Responder cannot directly resolve a ticket, the ticket SHOULD be included in a RESOLVE_INIT message sent to a KMS. If the Responder does not have a shared credential with the KMS that issued the ticket (KMS I) or if the Responder does not know which KMS issued the ticket, the Responder SHOULD send the RESOLVE_INIT message to one of the Responder’s trusted KMSs (KMS R). If KMS R did not issue the ticket, KMS R would normally be unable to directly resolve the ticket and must hence ask another KMS to resolve it (typically the issuing KMS).

The signaling between different KMSs MAY be done with a Ticket Resolve exchange as illustrated in Figure 9. The IDRr and TICKET payloads from the previous RESOLVE_INIT message SHOULD be reused. Note that IDRr cannot be used to look up the pre-shared key/ certificate.

11. Adding New Ticket Types to MIKEY-TICKET

The Ticket Data (in the TICKET payload) could be a reference to information (keys, etc.) stored by the key management service, it could contain all the information itself, or it could be a combination of the two alternatives. For systems serving many users, it is not ideal to use the reference-only ticket approach as this would force the key management service to keep state of all issued tickets that are still valid. Tickets may carry many different types of information helping to enforce usage policies. The policies may be group policies or per-user policies.

Tickets may either be transparent, meaning they can be resolved without contacting the KMS that generated them, or opaque, meaning that the original KMS must be contacted. The ticket information SHOULD typically be integrity protected and certain fields need confidentiality protection, in particular, the keys if explicitly included. Other types of information may also require confidentiality protection due to privacy reasons. In mode 2 (see Section 4.1.1), it may be preferable to include several encrypted ticket protection keys (similar to Secure/Multipurpose Internet Mail Extensions (S/MIME)) as this may allow multiple peers to resolve the ticket.

The Ticket Data MUST include information so that the resolving party can retrieve an encoded KEMAC. It MUST also be possible to verify the integrity of the TICKET payload. It is RECOMMENDED that future specifications use the recommended payload order and do not add any additional payloads or processing. New Ticket Types SHOULD NOT change the processing for the Responder. If a new Ticket Type requires additional processing, it MUST be indicated in the Ticket Policy (N and O flags). New specifications MUST specify which modes are supported and if any additional security considerations apply.

12. Security Considerations

Unless otherwise stated, the security considerations in [RFC3830] still apply and contain notes on the security properties of the MIKEY protocol, key derivation functions, and other components. As some security properties depend on the specific Ticket Type, only generic security considerations concerning the MIKEY-TICKET framework are discussed.

This specification includes a large number of optional features, which adds complexity to the general case. Protocol designers are strongly encouraged to establish strict profiles defining MIKEY-TICKET options (e.g., exchanges or message fields) that SHOULD or MUST be supported. Such profiles should preclude unexpected consequences from compliant implementations with wildly differing option sets.

12.1. General

In addition to the Ticket Policy, the KMS MAY have its own set of policies (authorized key lengths, algorithms, etc.) that in some way are shared with the peers. The KMS MAY also provide keying material to authorized intermediate nodes performing various network functions (e.g., transcoding services, recording services, conference bridges). The key management service can enforce end-to-end security by only distributing the keys to authorized end-users. As in [RFC3830], the user identities are not confidentiality protected. If user privacy is needed, some kind of Privacy Enhancing Technologies (PET) like anonymous or temporary credentials MAY be used.

In the standard MIKEY modes [RFC3830], the keys are generated by the Initiator (or by both peers in the Diffie-Hellman scheme). If a bad PRNG (Pseudorandom Number Generator) is used, this is likely to make any key management protocol sensitive to different kinds of attacks, and MIKEY is no exception. As the choice of the PRNG is implementation specific, the easiest (and often bad) choice is to use the PRNG supplied by the operating system. In MIKEY-TICKET's default mode of operation, the key generation is mostly done by the KMS, which can be assumed to be less likely to use a bad random number generator. All keys (including keys used to protect the ticket) MUST have adequate strength/length, i.e., 128 bits or more.

The use of random nonces (RANDs) in the key derivation is of utmost importance to counter offline pre-computation attacks and other generic attacks. A key of length n, using RANDs of length r, has effective key entropy of (n + r) / 2 against a birthday attack. Therefore, the sum of the lengths of RANDRi and RANDRr MUST at least be equal to the size of the longest pre-shared key/envelope key/MPK/TGK/GTGK, RANDRkms MUST at least be as long as the longest MPKr/TGK, and the RAND in the MIKEY base ticket MUST at least be as long as the longest of TPK and MPK.

Note that the CSB Updating messages reuse the old RANDs. This means that the total effective key entropy (relative to pre-computation attacks) for k consecutive key updates, assuming the TGKs are each n bits long, is still no more than n bits. In other words, the time and memory needed by an attacker to get all k n-bit keys are proportional to 2^n. While this might seem like a defect, this is in practice (for all reasonable values of k) not better than brute force, which on average requires k * 2^(n-1) work (even if different RANDs would be used). A birthday attack would only require 2^(n/2) work, but would need access to 2^(n/2) sessions protected with equally many different keys using a single pair of RANDs. This is, for typical values of n, clearly totally infeasible. The success probability of such an attack can be controlled by limiting the number of updates correspondingly. As stated in [RFC3830], the fact that more than one key can be compromised in a single attack is inherent to any solution using secret- or public-key algorithms. An attacker always gets access to all the exchanged keys by doing an exhaustive search on the pre-shared key/envelope key/MPK. This requires 2^m work, where m is the effective size of the key.

As the Responder MAY generate a RAND, the Ticket Transfer exchange can provide mutual freshness guarantee for all derived keys.

The new algorithms PRF-HMAC-SHA-256, AES-CM-256, and HMAC-SHA-256-256 use 256-bit keys and offer a higher security level than the previously defined algorithms. If one of the 256-bit algorithms are supported, the other two algorithms SHALL also be supported. The 256-bit algorithms SHOULD be used together, and they SHALL NOT be mixed with algorithms using key sizes less than 256 bits. If session keys (TEK/TGK/GTGK) longer than 128 bits are used, 128-bit algorithms SHALL NOT be used.

12.2. Key Forking

In some situations, the TRANSFER_INIT message may be delivered to multiple endpoints. For example, when a Responder is registered on several devices (e.g., mobile phone, fixed phone, and computer) or when an invite is being made to addresses of the type

IT-support@example.com, a group of users where only one is supposed to answer. The Initiator may not even always know exactly who the authorized group members are. To prevent all forms of eavesdropping, entities other than the endpoint that answers MUST NOT get access to the session keys.

When key forking is not used, keys are accessible by everyone that can resolve the ticket. When key forking is used, some keys (MPKr and TGKs encoded in the ticket) are modified, making them cryptographically unique for each responder targeted by the forking. As only the Initiator and the KMS have access to the master TGKs, it is infeasible for anyone else to derive the session keys.

When key forking is used, some keys (MPKi and TEKs and GTGK encoded in the ticket) are still accessible by everyone that can resolve the ticket and should be used with this in mind. This also concerns session keys transferred in a KEMAC in the first TRANSFER_INIT (as they are protected with MPKi).

12.3. Denial of Service

This protocol is resistant to denial-of-service attacks against the KMS in the sense that it does not construct any state (at the key management protocol level) before it has authenticated the Initiator or Responder. Since the Responder, in general, cannot verify the validity of a TRANSFER_INIT message without first contacting the KMS, denial of service may be launched against the Responder and/or the KMS via the Responder. Typical prevention methods such as rate-limiting and ACL (Access Control List) capability SHOULD therefore be implemented in the KMS as well as the clients. If something in the signaling is suspicious, the Responder SHOULD abort before attempting a RESOLVE_INIT with the KMS. The types and amount of prevention needed depends on how critical the system is and may vary depending on the Ticket Type.

12.4. Replay

In a replay attack, an attacker may intercept and later retransmit the whole or part of a MIKEY message, attempting to trick the receiver (Responder or KMS) into undesired operations, e.g., leading to a lack of key freshness. MIKEY-TICKET implements several mechanisms to prevent and detect such attacks. Timestamps together with a replay cache efficiently stop the replay of entire MIKEY messages. Parts of the received messages (or their hashes) can be saved in the replay cache until their timestamp is outdated. To prevent replay attacks, the sender's (Initiator or Responder) and the receiver's (Responder or KMS) identity is always (explicitly or implicitly) included in the MAC/Signature calculation.

An attacker may also attempt to replay a ticket by inserting it into a new MIKEY message. A possible scenario is that Alice and Bob first communicate based on a ticket, which an attacker Mallory intercepts. Later, Mallory (acting as herself) invites Bob by inserting the ticket into her own TRANSFER_INIT message. If key forking is used, such replays will always be detected when Bob has resolved the ticket. If key forking is not used, such replays will be detected unless Mallory has knowledge of the MPKi. And if Mallory has knowledge of the MPKi (i.e., she is authorized to resolve the ticket) and key forking is not used, there is no attack. For the reasons explained above, it is RECOMMENDED to use key forking.

12.5. Group Key Management

In a group scenario, only authorized group members must have access to the keys. In some situation, the communication may be initiated by the Initiator using a group identity and the Initiator may not even know exactly who the authorized group members are. Moreover, group membership may change over time due to leaves/joins. In such a situation, it is foremost the responsibility of the KMS to reject ticket resolution requests from unauthorized responders, implying that the KMS needs to be able to map an individual's identity (carried in the RESOLVE_INIT message) to group membership (where the group identity is carried in the ticket).

As noted, reuse of tickets, which bypasses the KMS, is NOT RECOMMENDED when the Initiator is not fully ensured about group membership status.

13. Acknowledgements

The authors would like to thank Fredrik Ahlqvist, Rolf Blom, Yi Cheng, Lakshminath Dondeti, Vesa Lehtovirta, Fredrik Lindholm, Mats Naslund, Karl Norrman, Oscar Ohlsson, Brian Rosenberg, Bengt Sahlin, Wei Yinxing, and Zhu Yunwen for their support and valuable comments.

14. IANA Considerations

This document defines several new values for the namespaces Data Type, Next Payload, PRF func, CS ID map type, Encr alg, MAC alg, TS Type, ID Type, Hash func, Error no, and Key Data Type defined in [RFC3830]. The following IANA assignments were added to the MIKEY Payload registry (in parentheses is a reference to the table containing the registered values):

- o Data Type (see Table 6.1)
- o Next Payload (see Table 6.2)

- PRF func (see Table 6.3)
- CS ID map type (see Table 6.4)
- Encr alg (see Table 6.5)
- MAC alg (see Table 6.6)
- TS Type (see Table 6.7)
- ID Type (see Table 6.9)
- Hash func (see Table 6.11)
- Error no (see Table 6.13)
- Key Data Type (see Table 6.14)

The TR payload defines an 8-bit TS Role field for which IANA has created and will maintain a new namespace in the MIKEY Payload registry. Assignments consist of a TS Role name and its associated value. Values in the range 1-239 SHOULD be approved by the process of Specification Required, values in the range 240-254 are Reserved for Private Use, and the values 0 and 255 are Reserved according to [RFC5226]. The initial contents of the registry are as follows:

Value	TS Role
0	Reserved
1	Time of issue (TRi)
2	Start of validity period (TRs)
3	End of validity period (TRe)
4	Rekeying interval (TRr)
5-239	Unassigned
240-254	Reserved for Private Use
255	Reserved

The IDR payload defines an 8-bit ID Role field for which IANA has created and will maintain a new namespace in the MIKEY Payload registry. Assignments consist of an ID Role name and its associated value. Values in the range 1-239 SHOULD be approved by the process of Specification Required, values in the range 240-254 are Reserved for Private Use, and the values 0 and 255 are Reserved according to [RFC5226]. The initial contents of the registry are as follows:

Value	ID Role
0	Reserved
1	Initiator (IDRi)
2	Responder (IDRr)
3	KMS (IDRkms)
4	Pre-Shared Key (IDRpsk)
5	Application (IDRapp)
6-239	Unassigned
240-254	Reserved for Private Use
255	Reserved

The RANDR payload defines an 8-bit RAND Role field for which IANA has created and will maintain a new namespace in the MIKEY Payload registry. Assignments consist of a RAND Role name and its associated value. Values in the range 1-239 SHOULD be approved by the process of Specification Required, values in the range 240-254 are Reserved for Private Use, and the values 0 and 255 are Reserved according to [RFC5226]. The initial contents of the registry are as follows:

Value	RAND Role
0	Reserved
1	Initiator (RANDRi)
2	Responder (RANDRr)
3	KMS (RANDRkms)
4-239	Unassigned
240-254	Reserved for Private Use
255	Reserved

The TP/TICKET payload defines a 16-bit Ticket Type field for which IANA has created and will maintain a new namespace in the MIKEY Payload registry. Assignments consist of a Ticket Type name and its associated value. Values in the range 1-61439 SHOULD be approved by the process of Specification Required, values in the range 61440-65534 are Reserved for Private Use, and the values 0 and 65535 are Reserved according to [RFC5226]. The initial contents of the registry are as follows:

Value	Ticket Type
0	Reserved
1	MIKEY base ticket
2	3GPP base ticket
3-61439	Unassigned
61440-65534	Reserved for Private Use
65535	Reserved

15. References

15.1. Normative References

[FIPS.180-3] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.

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

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

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 4330, January 2006.

[RFC4563] Carrara, E., Lehtovirta, V., and K. Norrman, "The Key ID Information Type for the General Extension Payload in Multimedia Internet KEYing (MIKEY)", RFC 4563, June 2006.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[RFC4738] Ignjatic, D., Dondeti, L., Audet, F., and P. Lin, "MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", RFC 4738, November 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

15.2. Informative References

[3GPP.33.328] 3GPP, "IP Multimedia Subsystem (IMS) media plane security", 3GPP TS 33.328 9.3.0, December 2010.

[Otway-Rees] Otway, D., and O. Rees, "Efficient and Timely Mutual Authentication", ACM SIGOPS Operating Systems Review v.21 n.1, p.8-10, January 1987.

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

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4650] Euchner, M., "HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY)", RFC 4650, September 2006.

[RFC5197] Fries, S. and D. Ignjatic, "On the Applicability of Various Multimedia Internet KEYing (MIKEY) Modes and Extensions", RFC 5197, June 2008.

[RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.

Appendix A. MIKEY Base Ticket

The MIKEY base ticket MAY be used in any of the modes described in Section 4.1.1. The Ticket Data SHALL be constructed of MIKEY payloads and SHALL be protected by a MAC based on a pre-shared Ticket Protection Key (TPK). The parties that shares the TPK depends on the mode. Unexpected payloads in the Ticket Data SHOULD be ignored.

```
              Ticket Data = THDR, T, RAND, KEMAC, [IDRpsk], V
```

A.1. Components of the Ticket Data

The Ticket Data MUST always begin with a Ticket Header payload (THDR). The ticket header is a new payload type; for the definition, see Appendix A.3.

T is a timestamp containing the time of issue or a counter. It MAY be used in the IV (Initialization Vector) formation (e.g., Section 4.2.3 of [RFC3830]).

RAND is used as input to the key derivation function when keys are derived from the TPK and the MPK (see Appendices A.2.1 and A.2.2).

The KEMAC payload SHALL use the NULL authentication algorithm, as a MAC is included in the V payload. The encryption key (encr_key) and salting key (salt_key) SHALL be derived from the TPK (see Appendix A.2.1). Depending on the encryption algorithm, the salting key be used in the IV creation (see Section 4.2.3 of [RFC3830]). If CSB ID is needed in the IV creation it SHALL be set to '0xFFFFFFFF'. The KEMAC is hence constructed as follows:

```
                 KEMAC = E(encr_key, MPK || {TEK|TGK|GTGK})
```

MPKi and MPKr are derived from the MPK as defined in Appendix A.2.2.

IDRpsk contains an identifier that enables the KMS/Responder to retrieve the TPK. It MAY be omitted when the TPK can be retrieved anyhow.

The last payload SHALL be a Verification payload (V) where the authentication key (auth_key) is derived from the TPK. The MAC SHALL be calculated over the entire TICKET payload except the Next Payload field (in the TICKET payload), the Initiator Data length field, the Initiator Data field, and the MAC field itself.

A.2. Key Derivation

The labels in the key derivations SHALL NOT include entire RAND payloads, only the fields RAND length and RAND from the corresponding payload.

A.2.1. Deriving Keys from a TPK

In the following, we describe how keying material is derived from a TPK. The key derivation method SHALL be executed using the PRF indicated in the Ticket Policy. The parameters for the PRF are:

```
   inkey:      : TPK
   inkey_len   : bit length of the inkey
   label       : constant || 0xFF || 0xFFFFFFFF || 0x05 ||
                 length RAND || RAND
   outkey_len : desired bit length of the outkey (encr_key,
                 auth_key, salt_key)
```

Length RAND is the length of RAND in bytes as an 8-bit unsigned integer. The constants are as defined in Section 4.1.4 of [RFC3830]. The key derivation and its dependencies on Ticket Data contents when AES-CM is used are illustrated in Figure 10. The key derivation is done by the party that creates the ticket (KMS or Initiator) and by the party that resolves the ticket (KMS or Responder). The encryption key and the IV are used to encrypt the KEMAC.

```
                                 -----          auth_key        ------
              -----     TPK     |     |----------------------->| AUTH |
             | TPK |----------->|     |       encr_key          ------
              -----             | PRF |--------------------+       |
                ^           +-->|     |     salt_key       |       |
                :           |   |     |----------------+   |       |
                :           |    -----                 |   |       |
                :           |                          v   |       |
       identify :      RAND |            TS value    ----  |       | MAC
                :           |         +------------>| IV | |       |
                :           |         |              ----  |       |
                :           |         |             IV |   |       |
                :           |         |                v   v       v
   Ticket    +---+----+---+--+---+---+-+-+------------+-------+---+---+
    Data     | IDRpsk |...| RAND |...| T |............| KEMAC |...| V |
             +--------+---+------+---+---+------------+-------+---+---+
```

Figure 10: Deriving keys from a TPK

A.2.2. Deriving MPKi and MPKr

In the following, we describe how MPKi and MPKr are derived from the MPK in the KEMAC payload. The key derivation method SHALL be executed using the PRF indicated in the Ticket Policy. The parameters for the PRF are:

```
inkey:      : MPK
inkey_len   : bit length of the inkey
label       : constant || 0xFF || 0xFFFFFFFF || 0x06 ||
              length RAND || RAND
outkey_len : desired bit length of the outkey (MPKi, MPKr)
              SHALL be equal to inkey_len
```

Length RAND is the length of RAND in bytes as an 8-bit unsigned integer. The constant depends on the derived key type as summarized below.

Derived key	Constant
MPKi	0x220E99A2
MPKr	0x1F4D675B

Table A.1: Constants for MPK key derivation

The constants are taken from the decimal digits of e as described in [RFC3830].

A.3. Ticket Header Payload (THDR)

The ticket header payload contains an indicator of the next payload as well as implementation-specific data.

```
 0                   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  !        THDR Data length       !   THDR Data   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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

* THDR Data (variable length): implementation specific data that SHOULD be ignored if it is not expected.

Appendix B. Alternative Use Cases

B.1. Compatibility Mode

MIKEY-TICKET can be used to define a Ticket Type compatible with [RFC3830] or any other half-round-trip key management protocol. The Initiator requests and gets a ticket from the KMS where the Ticket Data is a [RFC3830] message protected with a pre-shared key (KMS-Responder) or with the Responder's certificate. The Ticket Data is then sent to the Responder according to [RFC3830]. In this way, the Initiator can communicate with a Responder that only supports [RFC3830] and with whom the Initiator do not have any shared credentials.

```
+---+                          +-----+                          +---+
| I |                          | KMS |                          | R |
+---+                          +-----+                          +---+
            REQUEST_INIT
  -------------------------------->
            REQUEST_RESP
  <--------------------------------
                             3830 MIKEY
  ---------------------------------------------------------------->
```

Figure 11: Compatibility mode

Authors' Addresses

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
EMail: john.mattsson@ericsson.com

Tian Tian
ZTE Corporation
4F, RD Building 2, Zijinghua Road
Yuhuatai District, Nanjing 210012
P.R. China

Phone: +86-025-5287-7867
EMail: tian.tian1@zte.com.cn