3GPP TSG SA WG3 Security — S3#33 **S3-040235**
10 - 14 May 2004
Beijing, China

Agenda Item:	MBMS
Source:	Ericsson, Nokia
Title:	Extension payloads to MIKEY to support MBMS
Document for:	Discussion /Decision

1 Introduction

The MIKEY in S3-040081 [2] introduced unnecessary deviation from original MIKEY [1]. To minimize the impact on the MIKEY protocol, this document describes how the same information can be conveyed to the ME by means of the extension payload present in MIKEY. The purpose of this payload is, as the name suggests, to enable extensions to the protocol in a controlled fashion.

This document focuses on the case where all key handling is done in the ME.

2 The General Extension Payload

The general extension payload of MIKEY was defined to enable extensions to MIKEY in a controlled way, so that possible error conditions arising from non-supported extensions can be handled in a graceful way. The payload format has the following form:

Next Payload	**Type**	**Length**
Data		

where **Next Payload** is an 8-bit field identifying the payload after this one, **Type** is an 8-bit field specifying the type of data present in the **Data** field, and **Length** is a 16-bit field specifying the number of octets of the **Data** field.

The two values 0 and 1 are defined for the **Type** field in [1].

3 Usage of the General Extension Payload

The MIKEY in [2] introduced unnecessary deviation from original MIKEY [1] and can cause difficulties for implementations unaware of the extensions used in MBMS. The same information can be conveyed from the BM-SC to the ME by the use of the general extension payload. This approach has the benefit that legacy implementations can parse the messages unambiguously and discard unsupported ones.

Let the fields **Next Payload** and **Length** be as defined in [1] and the **Type** field be 2 (this should however be registered with IANA if time permits). The information needed for the combined method of key management in MBMS can now be carried in the **Data** field. The structure of the **Data** field is as follows:

Outer Key ID	Inner Key ID
MSK Fetch Point	

where **Outer Key ID** (16 bits) is the identity of the key used to encrypt the Key Data Sub Field in KEMAC and **Inner Key ID** is the identity of the key in the Key Data field in the Key Data Sub Field. If the key that is delivered is an MSK, there is also a **MSK Fetch Point** (see below) field after the Key ID:s. In the case there are more than one key in the KEMAC payload, there must be equally many ID:s in the extension payload. Furthermore, the order in which the keys are listed must correspond to the order of the list in the extension payload. In case there is not a one-to-one mapping between the two lists, the entire MIKEY message should be discarded.

The ME could use the Key ID:s to make sure that it does not try to install a key that is already in place. It is possible that the same key is delivered multiple times over the broadcast channel for reliability reasons. This would be seen as a replay attack (see Section 5), and it is also very inefficient to decrypt and authenticate a message with already known information. This of course puts the requirement on the ME to keep track of which keys are installed.

The **MSK Fetch Point** is intended to be used to tell the ME when it is time to pull a new MSK from the BM-SC. This field is added so that the network can spread the key requests from the ME:s to avoid congestion. It is not a good idea to allow the clients to chose the time for the pull, since it is likely that clients may chose a lazy strategy, where they pull at the end of or very close to the end of the lifetime of the key. Although there is no strong security in the proposed scheme, it has a higher probability of avoiding congestion by concurrent pulls from non-malicious but lazy ME:s. The **MSK Fetch Point** is specified in terms of MTK Key ID:s. That is, when the ME sees an MTK on the broadcast channel with the ID specified in the **MSK Fetch Point** field, it initiates a pull for a new MSK (the pull would also be initiated if the ME sees a MTK with a higher ID, which could happen if the packet containing the MTK with the ID specified in the MSK Fetch Point is lost on the radio link).

4 Usage of the KEMAC Payload

The KEMAC payload is used as specified in [1] with the exception of additional key types used in the Key Data Sub Field and the introduction of a new MAC algorithm identifier.

To distinguish the ``level´´ of the keys, the **Type** field is used. It has one of the following values (given in hexadecimal):

Key Type value	Meaning
0x04	The key in the **Key** field is an MSK and is encrypted with a MUK
0x05	The key in the **Key** field is an MTK and is encrypted with an MSK
0x06	The key in the **Key** field is an MSK for encryption and is encrypted with an MUK
0x07	The key in the **Key** field is an MSK for integrity protection and is encrypted with an MUK
0x08	The key in the **Key** field is an MTK for encryption and is encrypted with an MSK

0x09	The key in the **Key** field is an MTK for integrity protection and is encrypted with an MSK

The values 0x04 and 0x05 is to be used if key derivation (splitting of a key into one encryption and one integrity key) is to be done in the ME. Note that the idea of splitting keys is to create two or more cryptographically independent keys, where the compromise of one of them will not reveal any information about any of the other. The new type values should be registered with IANA if time allows it.

When the ME receives a MIKEY message containing an MSK, it uses a MUK_A (derived from MUK) to authenticate the message, and a MUK_E (derived from MUK) to decrypt the KEMAC payload.

When the ME receives a MIKEY message containing an MTK, it uses a MSK_A (derived from MSK or received as a separate key) to authenticate the message, and a MSK_E (derived from MSK or received separately) to decrypt the KEMAC payload.

An example of a (logical) MIKEY message that transports an MSK to the ME is shown in Figure 2. The MUK and MSK ID:s are left unencrypted, so that the ME can check if the MSK is already installed or not. Furthermore, the ME can read in the MSK Fetch Point field when it is time to pull another MSK from the network.

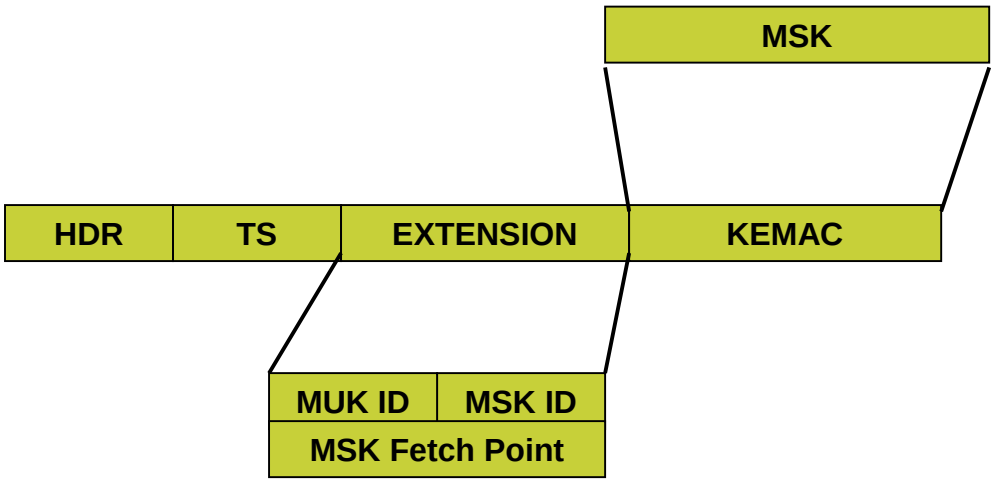

Figure 2. Example showing a (logical) MIKEY message transporting a MSK to the ME.

5 Replay Protection

The MIKEY message is protected for replay using the standard MIKEY mechanism by specifying the time stamp payload in [1] to be of type 2 (COUNTER).

6 Conclusions

The document gives a slightly different and more clean (with respect to MIKEY) way of achieving the same task as described in [2]. It describes how to use MIKEY in the case all key handling is done in the ME.

7 References

[1] Arkko et. al., ``draft-ietf-msec-mikey-08.txt´´, IETF draft, 2003, work in progress
[2] Nokia, ``S3-040081: Use of MIKEY in Combined method´´