# IEEE P802.21 Media Independent Handover Services

|  |
| --- |
| Detailed Proposal to IEEE 802.21d based on MKB for TGd |
| Date: 2013-01-06 |
| Author(s): |
| Name | Affiliation | Address | Phone | Email |
| Yoshikazu Hanatani | Toshiba | 1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582 Japan |  | yoshikazu.hanatani@toshiba.co.jp |
| Toru Kambayashi | Toshiba | 1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582 Japan |  | tooru.kamibayashi@toshiba.co.jp |
| Stephen Chasko | Landis+Gyr |  |  | Stephen.Chasko@landisgyr.com |
| Michael Demeter | Landis+Gyr |  |  | Michael.Demeter@landisgyr.com  |

Abstract

This proposal is a contribution for the 802.21d in response to 802.21-12-0091-06-MuGM-requirements-document. This proposal has two procedures: a group manipulation procedure based on Media Key Block (MKB) and a group command procedure.

Section numbering in this document corresponds to the one in IEEE 802.21-2008.

# Overview

# Normative references

***Insert the following references:***

TBD.

# Definitions

***Insert the following definitions in alphabetically order:***

**Group key block (GKB):**  A data that entities who have corresponding device keys can only decapsulate it and obtain a group key. See also: Annex P.

**Device key**: A data assigned to an entity in order to decapsulate a GKB.

**Media independent handover function identifier (MIHF ID):** An identifier to uniquely identify a single MIHF or a group of MIHFs.

**Individual media independent handover function identifier (Individual MIHF ID):** An MIHF ID to identify a single MIHF.

**Group media independent handover function identifier (Group MIHF ID):** An MIHF ID to identify a group of MIHFs.

**Group manipulation command:** A command to make members join in a group, update a group or leave from the group.

**Group command:** A command issued to members which belongs to a group via a multicast channel.

**Group Manager (GM):** An out-of-band server which generates a GKB.

**Command Center (CC):** A server which issues a group manipulation command and a group command. A CC resides in an MIH PoS.

# Abbreviations and acronyms

***Insert the following abbreviations and acronyms in alphabetically order:***

CC: Command Center

GM: Group Manager

GKB: Group Key Block

More?

# General architecture

## Introduction

## General design principles

## MIHF service overview

## Media independent handover reference framework

## MIHF reference models for link-layer technologies

## Service access points (SAPs)

## MIH protocol

# MIHF services

## General

## Service management

## Media independent event service

## Media independent command service

### Introduction

### Command service flow model

***Insert the following paragraph after the 1st paragraph:***

When a command request frame is sent to a group of MIHFs, it is transported using multicast transport and one or more remote MIHF(s) may receive the request frame and the local MIHF may receive one or more command response frame(s) from the remote MIHF(s). In this case, a CC who is an MIH User on an MIH PoS is the issuer of a group command and the MIH PoS is the sender of the group command request frame, and MN(s) or other MIH PoS(es) are the recipient of the group command request frame. Some group command requests do not require responses to be returned.

### Command lists

*Add the following entries to Table 7 (MIH Commands):*

|  |  |  |  |
| --- | --- | --- | --- |
| MIH command | (L) ocal / (R) emote | Comments | Defined in |
| MIH\_Configuration\_Update | R | This command is sent by PoS to a group of MNs or other PoSes to update their configuration. | 7.4.29 |
| MIH\_Group\_Manipulate | R | This command is sent by PoS to a group of MNs or other PoSes to create, delete or update a group. | 7.4.30 |
| MIH\_Push\_Certificate | R | This command is sent by PoS to a destination PoS or PoA | 7.4.31 |
| MIH\_Revoke\_Certificate | R | This commend is sent by PoS to a multicast group of PoS and/or PoA to revoke a certificate previously issued by the PoS. | 7.4.32 |

## Media independent information service

# Service access points (SAPs) and primitives

## Introduction

## SAPs

## MIH\_LINK\_SAP primitives

## MIH\_SAP primitives

### MIH\_Capability\_Discover

### MIH\_Register

#### MIH\_Register.request

#### MIH\_Register.indication

#### MIH\_Register.response

##### Semantics of service primitives

***Change the text as follows:***

MIH\_Register.response (

DestinationIdentifier,

Status,

ValidTimeInterval,

MulticastCipherSuite

)

***Add the following parameter:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| MulticastCipherSuite | MULTICAST\_CAP | (optional) Specifies the multicast ciphersuite to be used for securing multicast MIH message. Only one ciphersuite shall be included. |

#### MIH\_Register.confirm

##### Semantics of service primitives

***Change the text as follows:***

MIH\_Register.confirm (

SourceIdentifier,

Status,

ValidTimeInterval,

MulticastCipherSuite

)

***Add the following parameter:***

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| MulticastCipherSuite | MULTICAST\_CAP | (optional) Specifies the multicast ciphersuite to be used for securing multicast MIH message. Only one ciphersuite shall be included. |

### MIH\_DeRegister

### MIH\_Event\_Subscribe

### MIH\_Event\_Unsubscribe

### MIH\_Link\_Detected.indication

### MIH\_Link\_Up.indication

### MIH\_Link\_Down.indication

### MIH\_Link\_Parameters\_Report.indication

### MIH\_Link\_Going\_Down.indication

### MIH\_Link\_Handover\_Imminent.indication

### MIH\_Link\_Handover\_Complete.indication

### MIH\_Link\_PDU\_Transmit\_Status.indication

### MIH\_Link\_Get\_Parameters

### MIH\_Link\_Configure\_Thresholds

### MIH\_Link\_Actions

***Change section 7.4.16.1.2 as follows:***

MIH\_Link\_Actions.request (

Destination Identifier,

LinkActionsList,

GroupLinkActionsList

)

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| DestinationIdentifier | MIHF\_ID | This identifies the local MIHF or a remote MIHF that will be the destination of this request. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (optional) Specifies the suggested actions. This parameter shall be used if and only if DestinationIdentifier is an individual MIHF ID. |
| GroupLinkActionsList | LIST(GROUP\_LINK\_ACTION\_REQ) | (optional) Specifies the suggested actions. This parameter shall be used if and only if the DestinationIdentifier is a group MIHF ID. |

### MIH\_Net\_HO\_Candidate\_Query

### MIH\_MN\_HO\_Candidate\_Query

### MIH\_N2N\_HO\_Query\_Resources

### MIH\_MN\_HO\_Commit

### MIH\_Net\_HO\_Commit

***Change section 7.4.21.1.2 as follows:***

MIH\_Net\_HO\_Commit.request (

DestinationIdentifier,

LinkType,

TargetNetworkInfoList,

AssignedResourceSet,

LinkActionExecutionDelay,

LinkActionsList,

GroupLinkActionList

)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the MIHF ID of the MN MIHF (s) that is (are) to beCommitted. |
| LinkType | LINK\_TYPE | Contains target link type. |
| TargetNetworkInfoList | LIST(TGT\_NET\_INFO) | This list contains target network information for assisting the mobile node to perform a handover.  |
| AssignedResourceSet | ASGN\_RES\_SET | This includes the set of resource parameters assigned to the MN for performing the handover.  |
| LinkActionExecutionDelay | UNSIGNED\_INT(2) | Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out. This parameter shall be Used for non-group operation. |
| GroupLinkActionsList | LIST(GROUP\_LINK\_ACTION\_REQ) | (optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out. This parameter shall be Used for group operation. |

***Change section 7.4.21.2.2 as follows:***

MIH\_Net\_HO\_Commit.indication (

SourceIdentifier,

LinkType,

TargetNetworkInfoList,

AssignedResourceSet,

LinkActionExecutionDelay,

LinkActionsList,

GroupLinkActionsList

)

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** | **Description** |
| SourceIdentifier | MIHF\_ID | Specifies the MIHF ID of the node that sent the MIH\_Net\_HO\_Commit request message. |
| LinkType | LINK\_TYPE | Contains target link type. |
| TargetNetworkInfoList | LIST(TGT\_NET\_INFO) | This list contains target network information for assisting the mobile node to perform a handover.  |
| AssignedResourceSet | ASGN\_RES\_SET | This includes the set of resource parameters assigned to the MN for performing the handover.  |
| LinkActionExecutionDelay | UNSIGNED\_INT(2) | Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out. |
| LinkActionsList | LIST(LINK\_ACTION\_REQ) | (optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out.  |
| GroupLinkActionsList | LIST(GROUP\_LINK\_ACTION\_REQ) | (optional) Time (in ms) to elapse before an action needs to be taken. A value of 0 indicates that the action is taken immediately. Time elapsed is calculated from the instance the command arrives until the time when the execution of the action is carried out. This parameter shall be included if and only if LinkActionList is not used. |

### MIH\_N2N\_HO\_Commit

### MIH\_MN\_HO\_Complete

### MIH\_N2N\_HO\_Complete

### MIH\_Get\_Information

### MIH\_Push\_Information

### MIH\_Push\_Key\*

### MIH\_LL\_Auth\*

***Insert section 7.4.29 and 7.4.30 as follows:***

### MIH\_Configuration\_Update

#### MIH\_Configuration\_Update.request

##### Function

This primitive is generated by a PoS to update the configuration of one or more MN(s) or other PoS(es).

##### Semantics of service primitive

MIH\_Configuration\_Update.request (

DestinationIdentifier,

ConfigurationData)

|  |  |  |
| --- | --- | --- |
| Name | Data type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies MIHF-ID of the remote MIHF(s) to be configured. |
| ConfigurationData | OCTET\_STRING | Configuration data. |

##### When generated

The MIH user generates this primitive to update the configuration of one or more MN(s) and/or other PoS(es).

##### Effect on receipt

Upon receipt of this primitive, MIHF on the PoS sends the corresponding MIH\_Configuration\_Update indication message to the MN(s) or other PoS(es).

#### MIH\_Configuration\_Update.indication

##### Function

This primitive is generated by an MIHF to update the configuration of one or more MN(s) or other PoS(es).

##### Semantics of service primitive

MIH\_Configuration\_Update.indication (

SourceIdentifier,

GroupIdentifier,

ConfigurationData

)

|  |  |  |
| --- | --- | --- |
| Name | Data type | Descryption |
| SourceIdentifier | MIHF\_ID | Specifies MIHF-ID of the remote MIHF that sent MIH\_Configuration\_Update indication message. |
| GroupIdentifier | MIHF\_ID | Specifies the target group identifier. |
| ConfigurationData | OCTET\_STRING | Configuration data. |

##### When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Configuration\_Update indication message from a remote MIHF.

##### Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS may modify its configuration using the ConfigurationData parameter.

### MIH\_Group\_Manipulate

#### MIH\_Group\_Manipulate.request

##### Function

This primitive is generated by a PoS to manipulate group membership of one or more MN(s) or other PoS(es).

##### Semantics of service primitive

MIH\_Group\_Manipulate.request (

DestinationIdentifier,

GroupIdentifier,

GKBRange,

VerifyGroupKey,

AuxData,

CompleteSubtree,

GroupKeyData)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies group MIHF-ID of the remote MIHFs. DestinationIdentifier may be different from GroupIdentifier. |
| GroupIdentifier | MIHF\_ID, | The target group identifier for the group operation. |
| GKBRange | GKB\_Range | (optional) Valid range of GKB. |
| VerifyGroupKey | OCTET\_STRING | Verification data for group key. |
| AuxData | OCTET\_STRING | (optional) Auxiliary data. |
| CompleteSubtree | OCTET\_STRING | Complete Subtree data. |
| GroupKeyData | ENCR\_DATA | Encrypted group key. |

##### When generated

The MIH user generates this primitive to create, delete or modify a group.

##### Effect on receipt

Upon receipt of this primitive, MIHF on the PoS sends the corresponding MIH\_Group\_Manipulate indication message to the MN(s) or other PoS(es).

#### MIH\_Group\_Manipulate.indication

##### Function

This primitive is generated by an MIHF to manipulate group membership of a MN or a PoS.

##### Semantics of service primitive

MIH\_Group\_Manipulate.indication(

SourceIdentifier,

GroupIdentifier,

AuxData,

GroupStatus,

)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies MIHF-ID of the remote MIHF that issued MIH\_Group\_Manipulate.request. |
| GroupIdentifier | MIHF\_ID, | The target group identifier for the group operation.  |
| AuxData  | OCTET\_STRING | (optional) Auxiliary data. |
| GroupStatus | GROUP\_STATUS | Status of the group. |

##### When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Group\_Manipulate indication message from a remote MIHF.

##### Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS may join or leave a group specified in GroupIdentifier parameter. When the MIH User may also decrypt and install an encrypted group key that is associated with the specific group and contained in the GroupKeyData. The detailed procedure is described in 9.4.

### MIH\_Push\_Certificate

#### MIH\_Push\_Certificate.request

##### Function

This primitive is generated by a PoS used to send a certificate from a PoS to a destination PoS or MN.

##### Semantics of service primitive

MIH\_Push\_Certificate.request (

DestinationIdentifier,

Certificate)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the recipient of the certificate. |
| Certificate | CERTIFICATE | X.509 certificate |

##### When generated

A PoS generates this primitive for initial provisioning of certificates or for certificate updates.

##### Effect on receipt

Upon receipt of this primitive, MIHF on the sends the corresponding MIH\_Push\_Certificate request message to the destination MN or PoS.

#### MIH\_Push\_Certificate.indication

##### Function

This primitive is generated by a MIHF that receives an MIH\_Push\_Certificate indication message to manipulate group membership of one or more MN(s) or other PoS(es).

##### Semantics of service primitive

MIH\_Push\_Certificate.indication (

SourceIdentifier,

Certificate)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the sender of the certificate. |
| Certificate | CERTIFICATE | X.509 certificate |

##### When generated

The MIH user generates this primitive to create, delete or modify a group.

##### Effect on receipt

Certificate signature is verified and result of verification is provided back to push requester. After verification, validated certificate public keys within their expiration period can be utilized for multicast message.

#### MIH\_Push\_Certificate.response

##### Function

This primitive is generated by an MIH User to acknowledge receipt of a certificate from a PoS.

##### Semantics of service primitive

MIH\_Push\_Certificate.response (

DestinationIdentifier,

CertificateStatus)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the requestor of the certificate revocation. |
| CertificateSerialNumber | CERTIFICATE\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |
| CertificateStatus | CERT\_STATUS | Indicates whether a certificate has been verified and is now in use by the recipient. |

##### When generated

An MIH User generates this primitive after receipt and processing of certificate.

##### Effect on receipt

If certificate signature is valid, then MIH\_Push\_Certificate response message is sent back to the push requester. Result of request is provided in the REVOCATION\_STATUS.

#### MIH\_Push\_Certificate.confirm

##### Function

This primitive is generated by a MIHF that receives an MIH\_Push\_Certificate response to indicate the status of the certificate inspection.

##### Semantics of service primitive

MIH\_Push\_Certificate.confirm (

SourceIdentifier,

CertificateSerialNumber,

CertificateStatus)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the remote MIHF that invoked MIH\_Revoke\_Certificate.response. |
| CertificateSerialNumber | CERT\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |
| CertificateStatus | CERT\_STATUS | Indicates whether a certificate has been verified and is now in use by the recipient. |

##### When generated

The MIHF that receives an MIH\_Push\_Certificate response message generates this primitive to indicate the status of the certificate inspection.

##### Effect on receipt

If Certificate Status indicates success indicates that the PoS can manage device as being capable of received signed multicast messages.

### MIH\_Revoke\_Certificate

#### MIH\_Revoke\_Certificate.request

##### Function

This primitive is generated by a PoS used to revoke a certificate.

##### Semantics of service primitive

MIH\_Revoke\_Certificate.request (

DestinationIdentifier,

CertificateSerialNumber)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies an MIHF or a group of MIHFs to revoke the certificate. |
| CertificateSerialNumber | CERT\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |

##### When generated

The MIH user generates this primitive to revoke a certificate.

##### Effect on receipt

Upon receipt of this primitive, MIHF on the PoS sends the corresponding MIH\_Revoke\_Certificate request message to the destination MIHF(s).

#### MIH\_Revoke\_Certificate.indication

##### Function

This primitive is generated by an MIHF to revoke a certificate stored in MN(s) and PoS(es).

##### Semantics of service primitive

MIH\_Revoke\_Certificate.indication (

SourceIdentifier,

CertificateSerialNumber)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the remote MIHF that invoked MIH\_Revoke\_Certificate.request primitive. |
| CertificateSerialNumber | CERT\_SERIAL\_NUMBER | X.509 certificate subfield – serial number |

##### When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Revoke\_Certificate request message from a remote MIHF.

##### Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS deprecate the certificate specified by the CertificateSerialNumber and invokes a MIH\_Revoke\_Certificate.confirm primitive.

#### MIH\_Revoke\_Certificate.response

##### Function

This primitive is generated by an MIH User to acknowledge receipt of a certificate revocation request from a PoS.

##### Semantics of service primitive

MIH\_Revoke\_Certificate.response (

DestinationIdentifier,

CertificateStatus)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Specifies the remote MIHF that invoked MIH\_Revoke\_Certificate.request primitive. |
| Certificate Status | CERT\_STATUS | Indicates whether a certificate has been revoked. |

##### When generated

This primitive is generated by an MIHF on a MN or a PoS when receiving an MIH\_Revoke\_Certificate request message from a remote MIHF.

##### Effect on receipt

Upon receipt of this primitive, an MIH user on a MN or a PoS deprecate the certificate specified by the CertificateSerialNumber and invokes a MIH\_Revoke\_Certificate.confirm primitive.

#### MIH\_Revoke\_Certificate.confirm

##### Function

This primitive is generated by a MIHF that receives an MIH\_Revoke\_Certificate response to indicate the status of the certificate revocation.

##### Semantics of service primitive

MIH\_Revoke\_Certificate.confirm (

SourceIdentifier,

CertificateStatus)

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| SourceIdentifier | MIHF\_ID | Identifies the remote MIHF that invoked MIH\_Revoke\_Certificate.response. |
| Certificate Status | CERT\_STATUS | Indicates whether a certificate has been revoked. |

##### When generated

The MIHF that receives an MIH\_Revoke\_Certificate response message generates this primitive to indicate the status of the certificate revocation.

##### Effect on receipt

If Certificate Status indicates success for all the MIHFs to which certificate revocation request was sent, the PoS can changes status of the certificate to have been revoked.

## MIH\_NET\_SAP primitives

# Media independent handover protocol

## Introduction

## MIH protocol description

### MIH protocol transaction

### MIH protocol acknowledgement service

### MIH protocol transaction state diagram

### Other considerations

## MIH protocol identifiers

### MIHF ID

***Change section 8.3.1 as follows:***

MIHF Identifier (MIHF ID) is an identifier that is required to uniquely identify ~~an MIHF entity~~ a specific MIHF or a group of MIHFs for delivering the MIH services. MIHF ID is used in all MIH protocol messages. This enables the MIH protocol to be transport agnostic.

MIHF ID is assigned to the MIHF during its configuration process. The configuration process is outside the scope of the standard.

~~Multicast~~ Broadcast MIHF ID is defined as an MIHF ID of zero length. A ~~multicast~~ broadcast MIHF ID can be used when destination MIHF ID is not known to a source MIHF. The MIHF ID is of type MIHF\_ID. (See F.3.11.)

Multicast MIHF ID is defined as an MIHF ID with the specific prefix “\_G\_”. For example, *\_G\_sensornodes\_area\_A@foo.bar*is a multicast MIHF ID.

Other MIHF IDs are used for individual MIHF IDs.

### Transaction ID

## MIH protocol frame format

### General frame format

#### 8.4.1a Protected MIH protocol frame format

***Change section 8.4.1a as follows:***

In an MIH header the following two bits are used to indicate that an MIH PDU is protected and/or is used to carry a proactive authentication message.

a) P bit — Setting P bit to one indicates that the message carries a proactive authentication message.

b) S bit — Setting S bit to one indicates that an MIH security association exists and the service specific TLVs are protected.

A protected MIH PDU is an MIH PDU that has an MIH header with S bit set to one indicating that the MIH service specific TLVs in this PDU are ~~protected~~ encrypted or the PDU is digitally signed. When the MIH service specific TLVs in this PDU are encrypted, e~~E~~ach security association is defined for a pair of MIHF identifiers and is identified by a security association identifier (SAID). Therefore, for a protected MIH PDU, when a security association identifier is defined and the PDU is not digitally signed, the Source and Destination MIHF identifier TLVs may not be present. In this case, an MIH header is followed by an SAID TLV, which is followed by a security TLV. When no SAID TLV is carried, Service Specific TLVs shall be carried without encryption and therefore no Security TLV is carried. A Signature TLV is carried when a multicast PDU is digitally signed. When an MIH message with the S bit is set is multicast, both Source and Destination Identifier TLVs shall be carried even if a SAID TLV is carried.

Figure 28a shows a protected MIH protocol frame, where the source and destination MIHF TLVs and Signature TLV are optional.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MIH header(S=1) | Source MIHF Identifier TLV | Destination MIHF Identifier TLV | SAID TLV | Security TLV or Service Specific TLVs | Signature TLV |

**Figure 28a—Protected MIH frame format**

#### 8.4.1a.1 MIH PDU protected by (D)TLS

***Add the following text:***

A Signature TLV shall not be carried when MIH PDU is protected by (D)TLS.

#### 8.4.1a.2 MIH PDU protected through EAP-generated MIH SA

***Add the following text:***

A Signature TLV shall not be carried when MIH PDU is protected through EAP-generated MIH SA.

***Add the following section:***

#### 8.4.1a.3 MIH PDU protected through GKB-generated MIH SA

A group MIH security association (SA) may be established through GKB. A group MIH SA is established among a group of MIHFs. It includes a ciphersuite used for the protection. A security association identifier is assigned by the PoS as a result of successful GKB procedure. Figure 28e shows a protected MIH PDU for GKB-generated MIH SA without a Signature TLV. Figure 28f shows a protected MIH PDU for GKB-generated MIH SA with a Signature TLV. The protection procedure is specified in 9.4.



**Figure 28e: MIH PDU Protected by a GKB-generated MIH SA without a Signature TLV**

****

**Figure 28f: MIH PDU Protected by a GKB-generated MIH SA with a Signature TLV**

#### 8.4.1a.4 Protected MIH PDU upon transport address change

### Fragmentation and reassembly

## Message parameter TLV encoding

## MIH protocol messages

### MIH messages for service management

### MIH messages for event service

### MIH messages for command service

#### MIH\_Link\_Get\_Parameters request

#### MIH\_Link\_Get\_Parameters response

#### MIH\_Link\_Configure\_Thresholds request

#### MIH\_Link\_Configure\_Thresholds response

#### MIH\_Link\_Actions request

***Change the message format as follows:***

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=1, AID=3)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkActionsList (optional)(Link actions list TLV) |
| GroupLinkActionsList (optional)(Group Link actions list TLV) |

#### MIH\_Link\_Actions response

#### MIH\_Net\_HO\_Candidate\_Query request

#### MIH\_Net\_HO\_Candidate\_Query response

#### MIH\_MN\_HO\_Candidate\_Query request

#### MIH\_MN\_HO\_Candidate\_Query response

#### MIH\_N2N\_HO\_Query\_Resources request

#### MIH\_N2N\_HO\_Query\_Resources response

#### MIH\_MN\_HO\_Commit request

#### MIH\_MN\_HO\_Commit response

#### MIH\_Net\_HO\_Commit request

***Change the message format as follows:***

|  |
| --- |
| MIH Header Fields (SID=3, Opcode=1, AID=8) |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| LinkType(Link type TLV) |
| TargetNetworkInfoList(List of target network info TLV) |
| AssignedResourceSet(Assigned resource set TLV) |
| Link Action Execution Delay(Time interval TLV) |
| LinkActionsList (Optional)(Link actions list TLV) |
| GroupLinkActionsList (optional)(Group Link actions list TLV) |

#### MIH\_Net\_HO\_Commit response

#### MIH\_N2N\_HO\_Commit request

#### MIH\_N2N\_HO\_Commit response

#### MIH\_MN\_HO\_Complete request

#### MIH\_MN\_HO\_Complete response

#### MIH\_N2N\_HO\_Complete request

#### MIH\_N2N\_HO\_Complete response

#### Insert the following sections 8.6.3.23-8.6.3.28:

#### MIH\_Configuration\_Update indication

The corresponding MIH primitive of this message is defined in 7.4.29.

This message is used by the MIHF to change configuration of the MIH node(s) identified by the Destination Identifier.

The Destination Identifier is passed to the local MIH User as a GroupIdentifier in a MIH\_Configuration\_Update.indication.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=3, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| ConfigurationData(Configuration Data TLV) |

#### MIH\_Group\_Manipulate indication

The corresponding MIH primitive of this message is defined in 7.4.30.

This message is used by the MIHF to manipulate group membership of MIH node(s) identified by the Destination Identifier.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=3, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| GroupIdentifier(Group Identiifer TLV) |
| MulticastAddress(Multicast Address TLV) |
| GKBRange (optinal)(GKB Range TLV) |
| VerifyGroupKey(Verify Group Key TLV) |
| AuxData (optional)(Aux Data TLV) |
| CompleteSubtree(Complete Subtree TLV) |
| GroupKeyData(Group Key Data TLV) |

#### MIH\_Push\_Certificate request

The corresponding MIH primitive of this message is defined in 7.4.31.

This message is used by the MIHF to install a certificate to the MIH node identified by the Destination Identifier.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=1, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| Certificate(Certificate TLV) |

#### MIH\_Push\_Certificate response

The corresponding MIH primitive of this message is defined in 7.4.31.

This message is used by the MIHF to acknowledge receipt of a certificate from a PoS.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=2, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| CertificateSerialNumber(Certificate Serial Number TLV) |
|   |

#### MIH\_Revoke\_Certificate request

The corresponding MIH primitive of this message is defined in 7.4.32.

This message is used by the MIHF to revoke a certificate.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=1, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| CertificateSerialNumber(Certificate Serial Number TLV) |

#### MIH\_Revoke\_Certificate response

The corresponding MIH primitive of this message is defined in 7.4.32.

This message is used by the MIHF to acknowledge receipt of a certificate revocation request from a PoS.

|  |
| --- |
| **MIH Header Fields (SID=3, Opcode=2, AID=XX)** |
| Source Identifier = sending MIHF ID(Source MIHF ID TLV) |
| Destination Identifier = receiving MIHF ID(Destination MIHF ID TLV) |
| CertificateStatus(Certificate Status TLV) |

# MIH protocol protection\*

## Protection established through MIH (D)TLS

## Key establishment through an MIH service access authentication

## MIH message protection mechanisms for EAP-generated SAs

***Insert the following section:***

## Multicast MIH message protection mechanisms

### MIH message protection mechanisms for GKB-generated SAs

Introductory informative sections start here:

A Group Key Block (GKB) technology is used to manipulate groups of Mobile Nodes. A group manipulation command accompanies a target group ID and a GKB. The Mobile Nodes which receive a group manipulation command try to derive a group key from the GKB. If a Mobile Node succeeds to derive a group key, the Mobile Node will keep the pair of the target group ID and the group key, which means that the Mobile Node belongs to the group designated by the target group ID. Otherwise, that is, if a Mobile Node fails to derive a group key from the GKB, it means that the Mobile Node does not belong to the group designated by the target group ID. If the Mobile Node currently belongs to the group, it leaves the group: The Mobile Node renounces the corresponding pair of a group ID and a group key which it keeps.

A series of group commands may follow a group manipulation command which defines a target group of Mobile Nodes. A group command is issued, for instance, to instruct the group of Mobile Nodes that they should handover to a PoA or that they should update their configuration parameters. A payload of a group command may be encrypted using an SA derived from the group key. The MIH message protection mechanism based on GKB-generated SAs follows the two steps:

Step 1: A Command Center, which is an MIH PoS, issues a group manipulation command to instruct Mobile Nodes to join or leave a group. A group manipulation command may also be used to update a group key which Mobile Nodes keep. A group manipulation command may be delivered to Mobile Nodes through existing multicast channels. A multicast channel may be associated with a group: If a Mobile Node joins a group then it will listen to the multicast channel associated with the group.

Step 2: A Command Center issues to a group of Mobile Nodes a group command to instruct the Mobile Nodes in the group to take an action. The target group is designated by the group ID field in the group command. The group command may be delivered through the multicast channel associated with the group ID. A group command may simultaneously have two types of payload: Encrypted and non-encrypted. If a payload is encrypted, it is encrypted with a key derived from the current group key.

Each Mobile Node has a Device Key, which is a set of AES keys. The number of keys in a Device Key is 8, 16, 24 or 32, which is a common number in a system. Each Device Key accompanies a number called leaf number, the length of which is 8 bits, 16 bits, 24 bits or 32 bits, respectively, corresponding to the number of keys in a Device Key.

A Command Center is supposed to have a module called GKB Generator. A GKB Generator receives “virtually” all the Device Keys, a set of leaf numbers and a key. The set of leaf numbers indicates all the Mobile Nodes that constitute a group. The key is a (new) group key for that group. Note that a GKB Generator does not necessarily receive an entire Device Key tree. For instance, only a generator for a Device Key tree may be passed to it. Then the GKB Generator outputs a GKB, or several GKBs. The size of a GKB is limited and the GKB generator knows the maximum size. Before output, the GKB Generator appropriately divides a GKB into smaller GKBs if necessary.

While detailed procedures of an MIH User at a Command Center to prepare an MIH request for group manipulation, handover or configuration update depend on implementation of the MIH User, a rough sketch of the behaviors of the MIH User is provided at 9.4.2. The same section (i.e. 9.4.2) defines a series of actions to be performed by an MIHF of a Mobile Node which receives an indication of group manipulation, handover or configuration update. Those actions of a client MIHF are normative.

There are four modules involved in: The MIH User of a Command Center, the MIHF of a Command Center, the MIH User of a Mobile Node and the MIHF of a Mobile Node. Indispensable components for each of the modules relevant to group manipulation and group commands are listed as follows:

MIH User of Command Center:

* A GKB Generator
* All the Device Keys and the corresponding leaf numbers used in the Mobile Nodes. Note that a Device Key may be generated with the generator and a leaf number.
* A Group Management Database which stores a group management table, a row of which has the following four columns at least: A Group ID, a group key, a leaf number and an (Individual) MIHF ID. A Mobile Node with an MIHF ID in a row has the leaf number in the same row and belongs to the group designated by the Group ID recorded in the same row.

MIHF of Command Center:

* A signing key. The key is for creation of a signature of the Command Center.
* A Multicast Address Database which stores a multicast address table, a row of which has the following two columns at least: A Group ID and a multicast address. The multicast address in a row is associated with the group designated by the Group ID recorded in the same row.

MIHF of Mobile Node:

* A Device Key and the corresponding leaf number.
* A verification key. The key is for verification of a signature made by a Command Center.
* A Group Database which stores a group table, a row of which has the following three columns at least: A Group ID, a group key and a multicast address. The Mobile Node belongs to the group designated by the group ID in a row. The group key for the group is the one recorded in the same row, and the multicast address recorded in the same row is associated with the group.

### Secure group manipulation with group key distribution

Describe how a group is created, modified and deleted, with group key creation, update and revocation.

Fig 9.4.2 illustrates the group manipulation command distribution initiated by the command center via a multicast channel. When an MIH User of a command center issues a group manipulation command, it generates a MIH\_Group\_Manipulate.request described in 7.4.30.1. And then the MIH User passes the request to the MIHF of the command center. Upon receiving the request, MIHF generates MIH\_Group\_Manipulate.indication described in 8.6.3.24 and send it to MNs via multicast channels. When an MN receives the MIH\_Group\_Manipulate indication message, the MIHF of the MN processes the message. After processing the message, the MIHF sends MIH\_Group\_Manipulate.indication to the MIH User of the MN.



 Fig: 9.4.2

A MIH User of a command center generates MIH\_Group\_Manipulate.request described in 7.4.30.1 as follows:

1. Decide a group to manipulate. If it is a new group, choose a GroupIdentifer which is not currently used consulting with the Group Management Database. Then, decide members, i.e. MNs, of the group and a group key for the group. If the manipulating group is current one, there may be current members to be removed from the group.
2. Send to the GKB Generator all the Device Keys, the leaf numbers of the group members, and the group key. Then, the MIH User receives a GKB or a set of GKBs: A GKB contains a CompleteSubtree field, a GroupKeyData field and optionally a GKBRange field. A simple example how to make those fields is provided in Annex P. A GKB contains a GroupKeyData field if it is one of divided GKBs. Note that one MIH\_Group\_Manipulate.request contains one and only one GKB. Plural GKBs result in plural requests.
3. (Optional.) Define the AuxData field.
4. Define the DesitinationIdentifier. A DesitinationIdentifier represents an existing group associated with a multicast address. If one DestinationIdentifier does not suffice to cover the GKBRange, the GKBRange is changed to fit the DestinationIdentifier. And, other DestinationIdentifiers or a broadcast address is chosen to cover the original GKBRange (or the entire MNs if no GKBRange is originally provided). The Group Management Database serves the purpose to decide Destination Identifiers. Then, plural group manipulation commands are to be issued with the same CompleteSubtree field and the same GroupKeyData field.
5. Generate MIH\_Group\_Manipulate.request from the DestinationIdentifier, the GroupIdentifier, the GKBRange (an option), the VerifyGroupKey, the AuxData, the CompleteSubtree and GroupKeyData. Send it to the local MIHF.
6. Update the Group Management Database. If the target group of manipulation is an existing group then update it with new members and the new group key. If the target group is a new one, add a new group with new members and the new group key.

When the MIHF of the command center receives an MIH\_Group\_Manipulate.request from the MIH User, the MIHF generates and sends MIH\_Group\_Manipulate indications to appropriate multicast channels or to an broadcast channel:

1. Generate Source MIHF ID TLV using its own individual MIHF ID.
2. Generate Destination MIHF ID TLV based on the DestinationIdentifiner in the received MIH\_Group\_Mainpulate.request.
3. Generate Group Identifier TLV based on the GroupIdentifier in the received MIH\_Group\_Manipulation.request.
4. Generate Multicast Address TLV from the multicast address corresponding with the Destination Identifier in the received MIH\_Group\_Manipulate.request. The Multicast Address Database serves for the purpose to find the corresponding multicast address.
5. Generate GKB Range TLV based on the GKBRange in the received MIH\_Group\_Mainpulate.request.
6. Generate Verify Group Key TLV based on the VerifyGroupKey in the received MIH\_Group\_Manipulate.request.
7. Generate Aux Data TLV based on the AuxData in the received MIH\_Group\_Manipulate.request.
8. Generate Complete Subtree TLV based on the CompleteSubtree in the received MIH\_Group\_Manipulate.request.
9. Generate Group Key Data TLV from the GroupKeyData in the received MIH\_Group\_Manipulate.request.
10. Generate Signature TLV shown in 8.4.1a using the signing key of the command center.
11. Generate MIH\_Group\_Manipulate indications using the preceding TLVs and send MIH\_Group\_Manipulate indication message to the multicast address corresponding to the DestinationIdentifier.
12. Update Multicast Address Database if necessary. If the DesitinationIdentifer in the received request has already been registered in the database, obtain a multicast address associated with the DesitinationIdentifier anyhow and update the database with the new multicast address. The associated multicast address may be included in the request given by the MIH User. If the DestinationIdentifier in the received request is not registered in the database, obtain an associated multicast address anyhow and update the database with the new DestinationIdentifier and the new multicast address.

When a client receives the group manipulation command, the client’s MIHF processes the command as follows:

1. Extract a Source Identifier from the Source MIHF ID TLV.
2. Verify the Signature TLV using the verification key corresponding to the extracted Source Identifier. If the verification fails, cancel the following steps and stop processing.
3. Extract a Destination Identifier from the Destination MIHF ID TLV. Check if the Destination Identifier is registered or not as a Group ID in the Group Database. If it is not, cancel the following steps and stop processing.
4. If a GKBRange TLV exists in the indication, extract a GKBRange and check whether its own leaf ID is covered by the GKBRange or not. If it is not, cancel the following steps and stop processing.
5. Consider the Verify Group Key TLV, the Complete Subtree TLV and the Group Key Data TLV as a GKB, and process the GKB using its device key described in 9.4.2.1.2. If the process yields a group key, go to Step 6. Otherwise, go to Step 7.
6. Extract a GroupIdentifier from the Group Identifier TLV. If the GroupIdentifier is encrypted, decrypt it using the group key obtained in the previous step. Check whether the GroupIdentifier is registered or not as a Group ID in the Group Database. If it is, go to Step 8 [Update Group]. Otherwise, go to Step 9 [Join Group].
7. Extract a GroupIdentifier from the Group Identifier TLV. If the GroupIdentifier is encrypted, cancel the following steps and stop processing. Otherwise, check if the Group Database contains or not the GroupIdentifier as a Group ID. If it does, go to Step 10 [Leave Group]. Otherwise, just cancel the following steps and stop processing.
8. [Update Group] Obtain a multicast address associated with the GroupIdentifier anyhow. It may be obtained from a server. If the received indication has a Multicast Address TLV, check if it is encrypted or not. If it is encrypted, decrypt it using the new group key obtained from the GKB in Step 4. Update the Group Database where the record contains the GroupIdentifier, setting the new group key and the newly obtained multicast address. And, then, throw an MIH\_Group\_Manipulate.indication described in 7.4.30.2 to the MIH User in order to inform the MIH User of update of the group. The GroupStatus field is used for the purpose. If the received indication has an AuxData contained in the AuxData TLV, also throw the AuxData with the MIH\_Group\_Manipulation.indication. Cancel the following steps and stop processing.
9. [Join Group] Obtain a multicast address associated with the GroupIdentifier anyhow. It may be obtained from a server. If the received indication has a Multicast Address TLV, check if it is encrypted or not. If it is encrypted, decrypt it using the new group key obtained from the GKB in Step 4. Insert into the Group Database a record with the GroupIdentifier, the new group key and the multicast address. After that, throw an MIH\_Group\_Manipulation.indication described in 7.4.30.2 to the MIH User so that the MIH User may be informed of group participation. If the received indication has an AuxData contained in the AuxData TLV, also throw the AuxData with the MIH\_Group\_Manipulation.indication. Cancel the following steps and stop processing.
10. [Leave Group] Delete from the Group Database where the record contains the GroupIdentifier. After that, throw an MIH\_Group\_Manipulation.indication described in 7.4.30.2 to the MIH User so that the MIH User may know that it has left the group.

### GKB operation by the complete subtree method

A GKB is comprised of a Complete Subtree TLV, a Group Key Data TLV and a Verify Group Key TLV. A Complete Subtree TLV contains a list of node key IDs. A Group Key Data TLV contains ciphertexts of a group key encrypted by node keys. A node key used to decrypt each ciphertext in the Group Key Data TLV is specified by the Complete Subtree. A Verify Group Key TLV contains a MAC of a fixed message by the group key. It is used to check if a group key obtained from the Group Key Data TLV is correct or not.

### Encapsulation

Annex P provides an example of creation of a GKB at a GKB Generator.

### Decapsulation

An MN has a device key, which is a set of node keys. The followings are the GKB-decapsulation algorithm performed by an MN:

1. Find in the Complete Subtree TLV a node key ID which specifies a node key in the device key.
2. If there is no such a node key ID, the algorithm returns \bot, which is an error symbol and terminates. Otherwise, find a ciphertext in the Group Key Data TLV corresponding to the node key ID: If the node key ID is the i-th element of the Coplete Subtree TLV, the i-th element of the Group Key Data TLV is the ciphertext corresponding to the node key ID.
3. Decrypt the ciphertext using the node key specified by the node key ID. (See Step 1 above.) The result is a group key.
4. Check whether the group key is correct or not by checking the MAC in the Verify Group Key TLV. If the group key is correct, the algorithm returns the group key. Otherwise, the algorithm returns \bot.

### Multicast message encryption based on group key

Define group key hierarchy.



 Fig. 9.4.3a

When an MN successfully decupsulates a GKB, it obtains a group key. Three keys are derived from the group key. In this context, the group key may be called master group key (MGK). The keys derived from the MGK are a group integrity key (MIGIK) used to verify the MGK, a group manipulation encryption key (MIGMEK) used to protect a group manipulation command and a group encryption key (MIGEK) used to protect the group command. A type of deriving key is specified by a multicast ciphersuites described in 9.4.5.

A key derivation procedure is the same way described in 9.2.2

When a command center issues an MIH\_Configuration\_Update indication, the MIH User of the command center generates an MIH\_Configuration\_Update.request described in 7.4.29.1 and send it to the MIHF of the command center. The Configuration Data may be encrypted by the MIGEK derived from the MKG, which is the group key associated with the DesitinationIdentifier. The associated group key is found in the Group Management Database. Upon receiving the request, the MIHF of the command center behaves as follows:

1. Generate a Source MIHF ID TLV based on its own individual MIHF ID.
2. Generate a Destination MIHF ID TLV based on the DesitinationIdentifier in the received request.
3. Generate a Configuration Data TLV from the ConfigurationData in the received request.
4. Consulting with the Multicast Address Database, find the multicast address associated with the DesitinationIdentifer in the received request.
5. Generate an MIH\_Configuration\_Update indication described in 8.6.3.23, and send it to the multicast address found in Step 4.



Fig. 9.4.3b

When the MIHF of an MN receives a MIH\_Configuration\_Update indication, it throws an MIH\_Configuration\_Update.indication described in 7.4.29.2 to the MIH User of the MN:

1. Extract a Source Identifier from the Source MIHF ID TLV.
2. Verify the Signature TLV using the verification key corresponding with the extracted Source Identifier. If the verification fails, cancel the following steps and stop processing.
3. Extract a Destination Identifier from the Destination MIHF ID TLV and checks if the group designated by the Destination Identifier is registered in the Group Database. If it is not, cancel the following steps and stop processing.
4. Extract a ConfigurationData rom the Configuration Data TLV.
5. If the ConfigulationData is encrypted, decrypt it with the MIGEK derived from the MGK, where the MGK is the group key associated with the Destination Identifier. The group key is found in the Group Database.
6. Generate a Source Identifier TLV from the extracted Source Identifier.
7. Generate a GroupIdentifer TLV from the extracted Destination Identifer.
8. Generate a ConfigurationData TLV from the extracted ConfigurationData.
9. Generate a MIH\_Configuration\_Update.indication described in 7.4.29.2 and throw it to the MIH User.

### Signature and Certificate Management

In order to enable signing functionality, the message source requests certificates for public key using an out-of-band mechanism that is not specified in this specification. The message source provides the certificates to destination devices. Message signing procedure, signature verification procedure and certificate management procedure are described in 9.5.1, 9.5.2 and 9.5.3, respectively.

#### Multicast Message Signatures

Multicast Messages are signed with the message source using a private key of the message source. Integrity and proof of origin of a multicast message is verified by verifying the message signature with the public key of a message source.

On Receipt of signed multicast message there is an optional response indicating validity of signature. Message source requests certificates for key updates. Message source provides updates of certificates to destination devices (with overlap period).

The message content is signed using elliptical curve cryptography.

#### Signature Verification

The signature is verified using the message source signature verification key. The endpoints might have more than one key used for signature verification. This is to allow for key updates to happen in an efficient manner for large systems.

The message source will identify which key is to be used for the multicast message so that verification will utilize the correct key for signature verification.

#### Certificate Management

A root of trust will exist for the multicast nodes. The root of trust is envisioned to be a certificate authority. X.509 format certificates will be utilized. The root of trust will establish the binding between the identity of the message source and the public/private key pair used for signature generation and verification.

The certificate will include the identity of the certificate authority, the identity of the message source, the public key in use and the expiration date of the certificate and the certificate authority’s signature. For an endpoint to trust the certificate it must have the certificate authority public.

The initial certificates for multicast signature verification are distributed to multicast destinations as part of the provisioning process to the multi-node network. The certificates will include the certificate authority certificate used to verify the initial and updated certificates.

There will also be one or more certificates that are bound to the identity of the multicast source.

As part of the key update or revocation process, a new certificate will be provided to multicast destinations using the multicast mechanism. There needs to be a mechanism for multicast destinations to acknowledge the receipt of the multicast message.

When there is reduced trust in a certificate a mechanism will be provided to revoke the certificate from service. This mechanism will utilize the multicast messaging mechanism. Multicast destinations will need to provide a reply that indicates they have successfully revoked the certificate.

***Insert the following section.***

### Multicast Ciphersuites

The ciphersuites used for securing multicast MIH message is defined in Table xx.

Table 9.4.5: Multicast Ciphersuites

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Encryption Algorithm for Group Manipulation** | **Encryption Algorithm for Group Command** | **Digital Signature Algorithm** | **MAC Algorithm for Verify Group Key** |
| 10000000 | NULL | NULL | NULL | NULL |
| 10001000 | AES\_CBC-128 | AES\_CBC-128 | ECDSA-224 | AES\_CMAC-128 |
| 10001001 | AES\_CBC-128 | AES\_CBC-128 | ECDSA-256 | AES\_CMAC-128 |
| 10001100 | AES\_CBC-128 | NULL | ECDSA-224 | AES\_CMAC-128 |
| 10001101 | AES\_CBC-128 | NULL | ECDSA-256 | AES\_CMAC-128 |
| 10010000 | NULL | NULL | ECDSA-224 | NULL |
| 10010001 | NULL | NULL | ECDSA-256 | NULL |

## Common procedures *(originally section 9.4)*

### Sending *(originally section 9.4.1)*

When a PoS issues an MIH Service Specific TLV, the MIHF of the PoA generates a signature of the TLV using the signing key of the PoS and creates a Signature TLV from the generated signature.

### Receiving *(originally section 9.4.2)*

When an MN receives an MIH Specific TLV, the MIHF of the MN behaves as follows:

1. Verify the signature in the Signature TLV using the verification key corresponding the Source Identifier extracted from the received Source MIHF ID TLV. If the verification fails, cancel the following steps and stop processing.
2. Extract a Destination Identifier from the received Destination MIHF ID TLV. Check if the Destination Identifier is registered as a Group ID in the Group Database. If it is not, cancel the following steps and stop processing.
3. If a Security TLV exists in the MIH Specific TLV, decrypt the Security TLV using the MIGMEK derived from the MKG. The MGK is the group key corresponding to the Destination Identifier extracted in the previous step. The group key is found in the Group Database.

# Annex F Data type definition

### F.3.4 Data types for link identification and manipulation

***Insert the following data type:***

|  |  |  |
| --- | --- | --- |
| GROUP\_LINK\_ACTION\_REQ | SEQUENCE( LINK\_TYPE, CHOICE(NULL, LINK\_ADDR), LINK\_ACTION, LINK\_AC\_EX\_TIME) | A set of handover action request parameters. The choice of LINK\_ADDR is to provide PoA address information when the LINK\_ACTION contains the attribute for DATA\_FWD\_REQ.  |

### F.3.16 Data type for security

***Add a new ID\_TYPE enumeration defined in 802.21a-2012 as follows:***

|  |  |  |
| --- | --- | --- |
| ID\_TYPE | ENUMERATED | The type of security association.0: TLS-generated;1: EAP-generated2: GKB-generated |

***Add the following data types:***

|  |  |  |
| --- | --- | --- |
| SIGNATURE | OCTET\_STRING | A digital signature data. |
| MIH\_SEC\_CAP | SEQUENCE(TLS\_CAP,EAP\_CAP,MULTICAST\_CAP,) | Represents the MIH security capabilities. |
| MULTICAST\_CAP | UNSIGNED\_INT(2) | A multicast ciphersuite. Available multicast ciphersuites are defined in 9.6.  |
| CERTIFICATE | OCTET\_STRING | Provides a X.509 Certificate |
| CERT\_SERIAL\_NUMBER | OCTET\_STRING | Provides X.509 formatted certificate serial number which are unique by certificate authority. |
| CERT\_STATUS | ENUMERATED | This indicates the status of the certificate being pushed or revoked0 – Not Present – indicates that certificate is not present 1 – Certificate Valid – indicates that certificate is present and that associated public key is being used to verify signatures2 – Certificate Revoked3 - Certificate Expired |
| GKB\_RANGE | CHOICE( SEQUENCE( UNSIGNED\_INT(1), UNSIGNED\_INT(1)),SEQUENCE( UNSIGNED\_INT(2), UNSIGNED\_INT(2)),SEQUENCE( UNSIGNED\_INT(3), UNSIGNED\_INT(3)),SEQUENCE( UNSIGNED\_INT(4), UNSIGNED\_INT(4))) | A range of valid leaf identifiers in a complete subtree of a GKB. The first integer indicates the lowest value of the range. The second integer indicates the highest value of the range. |
| GROUP\_STATUS | ENUMERATED | This indicates a status of group manipulation command.0: Group has been added1: Group has been deleted2: Group has been updated. |

# Annex L MIH protocol message code assignment

***Allocate the following AIDs:***

|  |  |
| --- | --- |
| MIH messages | AID |
| MIH messages for Command Service |
| MIH\_Configuration\_Change | TBD |
| MIH\_Group\_Manpulate | TBD |
| MIH\_Push\_Certificate | TBD |
| MIH\_Revoke\_Certificate | TBD |

***Allocate the following TLV types:***

|  |  |  |
| --- | --- | --- |
| TLV type name | TLV type value | Data Type |
| Multicast Ciphersuite | TBD | MULTICAST\_CAP |
| Group Link actions list | TBD | GROUP\_LINK\_ACTION\_LIST |
| Configuration Data | TBD | OCTET\_STRING |
| Group Identiifer | TBD | CHOICE(MIHF\_ID, ENCR\_DATA) |
| Verify Group Key | TBD | OCTET\_STRING |
| Aux Data | TBD | OCTET\_STRING |
| Complete Subtree | TBD | OCTET\_STRING |
| Group Key Data | TBD | ENCR\_DATA  |
| Multicast Address | TBD | CHOICE(TRANSPORT\_ADDRESS, ENCR\_DATA) |
| GKB Range | TBD | GKB\_RANGE |
| Signature | TBD | SIGNATURE |
| Certificate | TBD | CERTIFICATE |
| Certificate Serial Number | TBD | CERT\_SERIAL\_NUMBER |
| Certificate Status | TBD | CERT\_STATUS |

# Annex P MKB Toy Example

TBD.