IEEE STANDARDS ASSOCIATION

IEEE Standard for

Local and metropolitan area networks—

# Part 21: Media Independent Handover Services

# Amendment 1: Security Extensions to Media Independent Handover Services and Protocol

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

3 May 2012

**IEEE Std 802.21a™-2012**
(Amendment to
IEEE Std 802.21™-2008)

**IEEE Std 802.21a™-2012**
(Amendment to
IEEE Std 802.21™-2008)

**IEEE Standard for**
**Local and metropolitan area networks—**

# Part 21: Media Independent Handover Services

# Amendment 1: Security Extensions to Media Independent Handover Services and Protocol

Sponsor

**LAN/MAN Standards Committee**
of the
**IEEE Computer Society**

Approved 29 March 2012

**IEEE-SA Standards Board**

**Abstract**: Extensions to IEEE Std 802.21-2008 are provided for security mechanisms to protect media independent handover services and mechanisms to use MIH to assist proactive authentication to reduce the latency due to media access authentication and key establishment with the target network.

**Keywords:** IEEE 802.21, IEEE 802.21a, proactive authentication, service access authentication, security protection

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA



**PDF: ISBN 978-0-7381-7257-6 STD97232**
**Print: ISBN 978-0-7381-7363-4 STDPD97232**



**Notice and Disclaimer of Liability Concerning the Use of IEEE Documents:** IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

**Translations:** The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

**Official Statements:** A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

**Comments on Standards:** Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important to ensure that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at http://standards.ieee.org/develop/wg/.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

**Photocopies:** Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

## Notice to users

### Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

### Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

### Updating of IEEE documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Website.

### Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

## Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

## Participants

At the time this amendment was submitted to the IEEE-SA for approval, the IEEE 802.21 Working Group had the following officers:

**Subir Das**, *Chair*
**Juan Carlos Zuniga**, *Vice-chair*
**David Cypher**, *Technical Editor*
**H. Anthony Chan**, *Secretary*

At the time this amendment was submitted to sponsor ballot, Security in Media Independent Handover task group had the following officers:

**Yoshihiro Ohba**, *Chair*
**Lidong Chen**, *Technical Editor*

When the IEEE 802.21 Working Group approved this amendment, the Working Group had the following membership:

Yoon Young An
Clint Chaplin
Lidong Chen
H. Anthony Chan
Subir Das
Antonio De la Oliva Delgado
Peretz Feder
Junghoon Jee
Hongseok Jeon
Lee Jin
Farrokh Khatibi
Dapeng Liu
Michael Lynch
Roger B. Marks
Paul Nikolich
Christian Niephaus
Yoshihiro Ohba
Changmin Park
Hyundo Park
Charlie Perkins
Ajay Rajkumar
Karen Randall
Stephen Shellhammer
Albert Vidal
Juan Carlos Zuniga

Major contributions were recieved from the following individuals:

Fernando Bernal-Hidalgo
Ashutosh Dutta
Toshikazu Kodama
Rafael Marin-Lopez

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

George Babut
Nancy Bravin
William Byrd
Radhakrishna Canchi
Anthony Chan
Clint Chaplin
Keith Chow
Charles Cook
Subir Das
Antonio De la Oliva Delgado
Thomas Dineen
Sourav Dutta
Richard Edgar
Pieter-Paul Giesberts
Reinhard Gloger
Ron Greenthaler
Randall Groves
Michael Gundlach
David Hunter
Ichirou Ida
Noriyuki Ikeuchi
Atsushi Ito
Junghoon Jee
Shinkyo Kaku
Stuart Kerry
Farrokh Khatibi
Brian Kiernan
Yongbum Kim
Bruce Kraemer
Joseph Kwak
Paul Lambert
Jan-Ray Liao
Greg Luri
Michael Lynch
Elvis Maculuba
Wayne Manges
Roger B. Marks
Gary Michel
Ronald Murias
Michael S. Newman
Yoshihiro Ohba
Satoshi Oyama
Venkatesha Prasad
Karen Randall
Maximilian Riegel
Robert Robinson
Benjamin Rolfe
Richard Roy
Randall Safier
Peter Saunderson
Bartien Sayogo
John Short
Gil Shultz
Kapil Sood
Thomas Starai
Rene Struik
Walter Struppler
Jun Ichi Takada
Steven Tilden
Mark-Rene Uchida
Dmitri Varsanofiev
Prabodh Varshney
John Vergis
Lei Wang
Stanley Wang
Hung-Yu Wei
Oren Yuen
Juan Carlos Zuniga

When the IEEE-SA Standards Board approved this standard on 29 March 2012, it had the following membership:

**Richard H. Hulett,** *Chair*
**John Kulick,** *Vice Chair*
**Robert M. Grow,** *Past Chair*
**Judith Gorman,** *Secretary*

Satish Aggarwal
Masayuki Ariyoshi
Peter Balma
William Bartley
Ted Burse
Clint Chaplin
Wael Diab
Jean-Philippe Faure
Alexander Gelman
Paul Houzé
Jim Hughes
Young Kyun Kim
Joseph L. Koepfinger*
David J. Law
Thomas Lee
Hung Ling
Oleg Logvinov
Ted Olsen
Gary Robinson
Jon Walter Rosdahl
Mike Seavey
Yatin Trivedi
Phil Winston
Yu Yuan

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Richard DeBlasio, DOE Representative
Michael Janezic, NIST Representative

Michelle Turner
*IEEE Standards Senior Program Manager, Document Development*

Lisa Perry
*IEEE Standards Program Manager, Technical Program Development*

# Introduction

This introduction is not part of IEEE Std 802.21-2012, IEEE Standard for Local and metropolitan area networks—Part 21: Media Independent Handover Service—Amendment 1: Security Extensions to Media Independent Handover Services and Protocol.

This amendment specifies the extensions to IEEE Std 802.21-2008 for security mechanisms to protect media independent handover services and mechanisms to use Media Independent Handover (MIH) to assist proactive authentications to reduce the latency due to media access authentication and key establishment with the target network.

# Contents

**IEEE Standard for**
**Local and metropolitan area networks—**

# Part 21: Media Independent Handover Services

# Amendment 1: Security Extensions to Media Independent Handover Services and Protocol



(This amendment specifies enhancements to IEEE Std 802.21™.)

NOTE—The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.[1]

[1]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

# 1. Overview

## 1.3 General

***Change the list in 1.3 as follows:***

The following items are not within the scope of this standard:

- Intra-technology handover [except for handovers across extended service sets (ESSs) in case of IEEE 802.11]
- Handover policy
- ~~Security mechanisms~~ Media specific protection mechanisms
- Enhancements specific to particular link-layer technologies that are required to support this standard (they will be carried out by those link-layer technology standards)
- Higher layer (layer 3 and above) enhancements that are required to support this standard

# 2. Normative references

***Insert the following normative references into Clause 2 in alphabetical order:***

FIPS 198, The Keyed-Hash Message Authentication Code (HMAC).[2]

IETF RFC 2409 (1998) The Internet Key Exchange (IKE).[3]

IETF RFC 3748 (2004) Extensible Authentication Protocol (EAP).

IETF RFC 4302 (2005), IP Authentication Header.

IETF RFC 4303 (2005), IP Encapsulating Security Payload.

IETF RFC 4306 (2005), The Internet Key Exchange (IKEv2) Protocol.

IETF RFC 4347 (2006), Datagram Transport Layer Protocol.

IETF RFC 5246 (2008), The Transport Layer Security (TLS) Protocol Version 1.2.

IETF RFC 5296 (2008), EAP Extensions for EAP Re-authentication Protocol (ERP).

NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation.[4]

NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation—The CMAC Mode for Authentication.

NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation—The CCM Mode for Confidentiality and Authentication.

NIST SP 800-108, Recommendation for Key Derivation Using Pseudorandom Functions.

[2]FIPS publications are made availble at: http://csrc.nist.gov/publications/PubsFIPS.html.

[3]IETF RFCs are available from the Internet Engineering Task Force website at http://www.ietf.org/rfc.html.

[4]NIST publications are available from the National Institute of Standards and Technology (http://www.nist.gov/).

## 3. Definitions

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

**authenticated encryption:** An algorithm to convert plaintext data to ciphertext and generate a message authentication code with a cryptographic key as a parameter to provide confidentiality, integrity, and authenticity of the data. *See also*: **encryption**; **MIC algorithm**.

**authentication process:** A process to assure that the claimed identity belongs to the entity. It is also called entity authentication. In this standard, an access authentication is an entity authentication with the identity used to access a specific network or a media independent handover (MIH) service.

**authentication server:** A server used for authentication purposes. When EAP is used as an authentication protocol, the authentication server is an EAP server.

**authenticator**: A network entity to execute EAP with a mobile node called a peer. An authenticator can use a backend server to conduct EAP execution. *Syn*: **EAP authenticator**.

**candidate authenticator**: An authenticator that is associated with a candidate PoA.

**candidate network**: A network that is a potential target to the MN's movement

**candidate PoS**: A potential PoS that can serve the MNs after movement.

**decryption**: An algorithm to convert ciphertext of data to plaintext with a cryptographic key as a parameter. It is an inverse operation of encryption.

**EAP authenticator**: *See*: **authenticator**.

**EAP peer**: The entity that responds to the EAP authenticator.

**EAP Re-authentication**: An authentication protocol using a key established in a previous EAP execution as defined in IETF RFC 5296.[5]

**EAP Server**: The entity that terminates the EAP execution with the EAP peer. In the case where no backend authentication server is used, the EAP server is a part of the EAP authenticator. In the case where a backend authentication server is used, the EAP server is located on the backend authentication server.

**encryption**: An algorithm to convert plaintext data to ciphertext to provide confidentiality with a cryptographic key as a parameter.

**extensible authentication protocol (EAP)**: An access authentication framework specified in IETF RFC 3748. It can support different authentication methods, called EAP methods.

**media specific authentication server**: An authentication server used for media specific access authentication.

**media specific authenticator**: An authenticator used for a media specific network access authentication.

[5]Information on normative references can be found in Clause 2.

**media specific network access authentication**: An authentication protocol for media access purpose specified for a specific media access. It may establish keys to be used in media specific protection mechanisms.

**media specific protection mechanism**: A mechanism that is applied to media specific layers to protect the data traffic using an encryption algorithm, an integrity protection algorithm, an authenticated encryption algorithm, or a combination of an encryption algorithm and an integrity protection algorithm.

**message authentication code (a.k.a. message integrity code)**: A data string generated over a message with a symmetric key by an algorithm, called message authentication code algorithm. It is used to verify the integrity of the message and to authenticate the origin of the message.

**message authentication code algorithm**: An algorithm to generate a message authentication code on a data message with a symmetric key to provide integrity protection and message origination authentication. *See:* **message authentication code.**

**message integrity code (MIC):** *See*: **message authentication code**.

**MIH security association (SA)**: An MIH security association is a set of cryptographic attributes established between the peer MIH entities for protecting MIH messages at the MIH protocol layer. An MIH SA is established via TLS handshake or EAP execution, where both the TLS handshake and EAP execution take place over the MIH protocol. When an MIH SA is established via TLS handshake, the TLS master key and its child keys, TLS random values and the TLS cipher suite negotiated in the TLS handshake are a part of the MIH SA. When an MIH SA is established via EAP execution, an MSK or rMSK and its child keys, MIH random values and the MIH cipher suite negotiated between the peer MIH entities are associated with the MIH SA.

**MIH service access authentication**: An authentication process that authorizes the access to media independent services.

**MIH service access authentication server**: An authentication server used to execute the MIH service access authentication. *See*: **authentication server**.

**proactive authentication**: A media specific authentication with the candidate network(s) executed prior to a handover to one of the candidate networks.

**protection mechanisms for MIH messages**: A protection mechanism that is applied to MIH PDU using an encryption algorithm, an integrity protection algorithm, an authenticated encryption algorithm, or a combination of an encryption algorithm and an integrity protection algorithm.

**serving authenticator**: The authenticator which is associated with the serving PoA.

**serving PoS**: An MIH PoS that is currently providing the MIH services to the mobile node.

**Security association identifier (SAID)**: An identifier of an MIH security association. When an SA is established through TLS, it is the TLS session ID. When an SA is generated through an EAP execution, it is assigned by the authenticator and the ID value is an octet string unique for a pair of MIH functions.

***Change definition 3.17 media independent handover point of service (MIH PoS) as follows:***

**3.17 media independent handover point of service (MIH PoS)**: Network-side MIHF instance that exchanges MIH messages with an MN-based MIHF. The same MIH Network Entity includes an MIH PoS for each MIH-enabled mobile node with which it exchanges MIH messages. A single MIH PoS can host more than one MIH service. The same MIH Network Entity can include multiple MIH Points of Service that

can provide different combinations of MIH services to the respective mobile nodes based on subscription or roaming conditions. Note that for a network entity comprising multiple interfaces, the notion of MIH PoS is associated with the network entity itself and not with just one of its interfaces. For MIH service access authentication, a PoS serves as an authenticator. Moreover, when a service access authentication establishes keys for proactive authentication, a PoS provides key distribution service for media specific authenticators.

# 4. Abbreviations and acronyms

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

| | |
|---|---|
| AES | advanced encryption standard |
| AS | authentication server |
| CBC | cipher block chaining |
| CCM | counter with CBC message authentication code |
| DTLS | datagram transport layer security |
| EAP | extensible authentication protocol |
| ERP | EAP Re-authentication Protocol |
| HMAC | keyed-hash message authentication code |
| IV | Initialization vector |
| MIAK | media independent authentication key |
| MIC | message integrity code |
| MIEK | media independent encryption key |
| MIIK | media independent integrity key |
| MISK | media independent session key |
| MSA | media specific authenticator |
| MSK | master session key |
| MSPMK | media specific pairwise master key |
| MSRK | media specific root key |
| PRF | pseudorandom function |
| rMSK | re-authentication master session key |
| SA | security association |
| SAID | security association identifier |
| SHA | secure hash algorithm |
| TLS | transport layer security |
| TLV | Type Length Value (a form of encoding, or an item encoded using that encoding) |

# 5. General architecture

## 5.1 Introduction

***Insert new 5.1.9 after 5.1.8 as follows:***

### 5.1.9 Proactive authentication and key establishment

This standard provides mechanisms for a mobile node to conduct a proactive authentication and key establishment with the candidate authenticator(s) and PoA(s). The proactive authentication is conducted through media specific network access authentication where authentication messages are exchanged between authentication end-points via a PoS. The MIH protocol is used for encapsulating the authentication messages between the MN and the PoS. A successful proactive authentication and key establishment allow a PoA in the target network to obtain a key(s) to protect the communication link between the mobile node and the PoA after the handover.

# 6. MIH service

## 6.2 Service management

### 6.2.1 General

***Change list after first paragraph as follows:***

Prior to providing the MIH services from one MIHF to another, the MIH entities need to be configured properly. This is done through the following service management functions:

— MIH capability discovery
— MIH registration
— MIH service access authentication
— MIH event subscription

### 6.2.2 Service management primitives

***Insert new rows after last row in Table 3 as follows:***

**Table 3—Service management primitives**

| Service management parameters | (L)ocal (R)emote | Defined in | Comments |
|---|---|---|---|
| MIH_Push_Key | L, R | 7.4.27 | Install a key in a remote PoA |
| MIH_LL_Auth | L, R | 7.4.28 | Carry out a proactive authentication over MIH between the MN and the PoS using link layer frames. |

## 6.5 Media independent event service

### 6.5.4 Information elements

***Insert information elements in Table 10 as follows:***

**Table 10—Information elelments**

| Name of information elements | Description | Data type |
|---|---|---|
| **PoA-specific information elements** | | |
| IE_AUTHENTICATOR_LINK_ADDR | An L2 address of the authenticator, which serves the PoA | LINK_ADDR |
| **PoA-specific higher layer service information elements** | | |
| IE_AUTHENTICATOR_IP_ADDR | The IP address of the authenticator, which serves the PoA. | IP_ADDR |
| IE_PoS_IP_ADDR | PoS's IP address. | IP_ADDR |

# 7. Service access point (SAP) and primitives

## 7.4 MIH_SAP primitives

### 7.4.1 MIH_Capability_Discover

#### 7.4.1.1 MIH_Capability_Discover.request

#### 7.4.1.1.2 Semantics of service primitives

***Change text as follows:***

MIH_Capability_Discover.request (
DestinationIdentifier,
LinkAddressList,
SupportedMihEventList,
SupportedMihCommandList,
SupportedIsQueryTypeList,
SupportedTransportList,
MBBHandoverSupport,
SupportedSecurityCapList
)

***Insert the following parameters:***

Parameters:

| Name | Data type | Description |
|---|---|---|
| SupportedSecurityCapList | MIH_SEC_CAP | (Optional) List of supported MIH security capabilities on the local MIHF. |

#### 7.4.1.2 MIH_Capability_Discover.indication

#### 7.4.1.2.2 Semantics of service primitive

***Change text as follows:***

MIH_Capability_Discover.indication (
SourceIdentifier,
LinkAddressList,
SupportedMihEventList,
SupportedMihCommandList,
SupportedIsQueryTypeList,
SupportedTransportList,
MBBHandoverSupport,
SupportedSecurityCapList
)

***Insert the following parameters:***

Parameters:

| Name | Data type | Description |
|---|---|---|
| SupportedSecurityCapList | MIH_SEC_CAP | (Optional) List of supported MIH security capabilities on the remote MIHF. |

### 7.4.1.3 MIH_Capability_Discover.response

### 7.4.1.3.2 Semantics of service primitive

***Change text as follows:***

MIH_Capability_Discover.response(
DestinationIdentifier,
Status,
LinkAddressList,
SupportedMihEventList,
SupportedMihCommandList,
SupportedIsQueryTypeList,
SupportedTransportList,
MBBHandoverSupport,
SupportedSecurityCapList
)

***Insert the following parameters:***

Parameters:

| Name | Data type | Description |
|---|---|---|
| SupportedSecurityCapList | MIH_SEC_CAP | (Optional) List of supported MIH security capabilities on the local MIHF. |

### 7.4.1.4 MIH_Capability_Discover.confirm

### 7.4.1.4.2 Semantics of service primitive

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

MIH_Capability_Discover.confirm(
SourceIdentifier,
Status,
LinkAddressList,
SupportedMihEventList,
SupportedMihCommandList,
SupportedIsQueryTypeList,
SupportedTransportList,
MBBHandoverSupport,
SupportedSecurityCapList
)

***Insert the following parameters:***

Parameters:

| Name | Data type | Description |
|---|---|---|
| SupportedSecurityCapList | MIH_SEC_CAP | (Optional) List of supported MIH security capabilities on the remote MIHF. |

### 7.4.17 MIH_Net_HO_Candidate_Query

### 7.4.17.2 MIH_Net_HO_Candidate_Query.request

### 7.4.17.2.2 Semantics of service primitive

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

MIH_Net_HO_Candidate.Query.request (
DestinationIdentifier,
SuggestedNewLinkList,
SuggestedCandidateAuthenticatorList,
QueryResourceReportFlag
)

***Insert the following parameters:***

Parameters

| Name | Data type | Description |
|---|---|---|
| SuggestedCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR_LIST) | List of media specific authenticator's addresses for the suggested candidate PoAs. |

### 7.4.17.3 MIH_Net_HO_Candidate_Query.indication

### 7.4.17.3.2 Semantics of service primitive

***Change text as follows:***

MIH_Net_HO_Candidate.Query.indication (
SourceIdentifier,
SuggestedNewLinkList,
SuggestedCandidateAuthenticatorList,
QueryResourceReportFlag
)

***Insert the following parameter:***

Parameters

| Name | Data type | Description |
|---|---|---|
| SuggestedCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR_LIST) | List of media specific authenticator's addresses for the suggested candidate PoAs. |

### 7.4.17.4 MIH_Net_HO_Candidate_Query.response

#### 7.4.17.4.1 Function

#### 7.4.17.4.2 Semantics of service primitive

***Change text as follows:***

MIH_Net_HO_Candidate.Query.response(
DestinationIdentifier,
Status,
SourceLinkIdentifier,
HandoverStatus,
PreferedLinkList,
PreferredCandidateAuthenticatorList
)

***Insert the following parameter:***

Parameter

| Name | Data type | Description |
|---|---|---|
| PreferredCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR_LIST) | List of the corresponding media specific authenticator's addresses for the preferred candidate PoAs. |

### 7.4.17.5 MIH_Net_HO_Candidate_Query.confirm

#### 7.4.17.5.2 Semantics of service primitive

***Change text as follows:***

MIH_Net_HO_Candidate.Query.confirm(
SourceIdentifier,
Status,
SourceLinkIdentifier,
HandoverStatus,
PreferedLinkList,
PreferredCandidateAuthenticatorList
)

***Insert the following parameter:***

Parameter

| Name | Data type | Description |
|---|---|---|
| PreferredCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR_LIST) | List of the corresponding media specific authenticator's addresses for the preferred candidate PoAs. |

### 7.4.18 MIH_MN_HO_Candidate_Query

### 7.4.18.1 MIH_MN_HO_Candidate_Query.request

### 7.4.18.1.2 Semantics of service primitive

***Change the following text as indicated as follows:***

MIH_MN_HO_Candidate.Query.request(
DestinationIdentifier,
SourceLinkIdentifier,
CandidateLinkList,
QoSResourceRequirements,
IPConfigurationMethods,
DHCPServerAddress,
FAAddress
AccessRouteAddress,
CandidateAuthenticatorList
)

***Insert the following parameter as follows:***

Parameter

| Name | Data type | Description |
|---|---|---|
| CandidateAuthenticatorList | LIST(LINK_AUTHENTICATOR_LIST) | List of the corresponding media specific authenticator's addresses for the candidate PoAs. |

### 7.4.18.2 MIH_MN_HO_Candidate_Query.indication

### 7.4.18.2.2 Semantics of service primitive

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

MIH_MN_HO_Candidate.Query.indication(
DestinationIdentifier,
SourceLinkIdentifier,
CandidateLinkList,
QoSResourceRequirements,
IPConfigurationMethods,
DHCPServerAddress,

FAAddress
AccessRouteAddress,
CandidateAuthenticatorList
)

***Insert the following parameter as follows:***

Parameter

| Name | Data type | Description |
|---|---|---|
| CandidateAuthenticatorList | LIST(LINK_AUTHENTICATOR_LIST) | List of the corresponding media specific authenticator's addresses for the candidate PoAs. |

### 7.4.18.3 MIH_MN_HO_Candidate_Query.response

### 7.4.18.3.2 Semantics of service primitive

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

MIH_MN_HO_Candidate_Query.response(
DestinationIdentifier,
Status,
SourceLinkIdentifier,
HandoverStatus,
PreferedCandidateLinkList,
PreferredCandidateAuthenticatorList
)

***Insert the following parameter:***

Parameter

| Name | Data type | Description |
|---|---|---|
| PreferredCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR_LIST) | List of the corresponding media specific authenticator's addresses for the preferred candidate PoAs. |

### 7.4.18.4 MIH_MN_HO_Candidate_Query.confirm

### 7.4.18.4.2 Semantics of service primitive

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

MIH_MN_HO_Candidate_Query.confirm(
DestinationIdentifier,
Status,
SourceLinkIdentifier,
HandoverStatus,
PreferedCandidateLinkList,
PreferredCandidateAuthenticatorList
)

***Insert the following parameter:***

Parameter

| Name | Data type | Description |
|---|---|---|
| PreferredCandidateAuthenticator-List | LIST(LINK_AUTHENTICATOR _LIST) | List of the corresponding media specific authenticator's address for the preferred candidate PoAs. |

***Insert new subclauses (7.4.27 and 7.4.28) as follows*:**

### 7.4.27 MIH_Push_Key

#### 7.4.27.1 MIH_Push_key.request

##### 7.4.27.1.1 Function

This primitive is used to request a remote MIHF (PoS) to install a key(s) in a target PoA(s).

##### 7.4.27.1.2 Semantics of service primitive

MIH_Push_key.request (
DestinationIdentifier,
LinkTupleIdentifierList
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| DestinationIdentifier | MIHF_ID | This identifies a remote MIHF that will be the destination of this request. |
| LinkTupleIdentifierList | LIST(LINK_TUPLE_ID) | This identifies a list of links of target PoAs for which keys are pushed. |

##### 7.4.27.1.3 When generated

This primitive is generated by an MIH user in the MN to request a remote MIHF in the serving PoS to install a key in a target PoA.

##### 7.4.27.1.4 Effect on receipt

The local MIHF shall generate an MIH_Push_Key request message to the remote MIHF.

#### 7.4.27.2 MIH_Push_key.indication

##### 7.4.27.2.1 Function

This primitive is used to pass a key to the corresponding MIH user on the serving PoS.

#### 7.4.27.2.2 Semantics of service primitive

MIH_Push_key.indication (
SourceIdentifier,
KeyMapping
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| SourceIdentifier | MIHF_ID | This identifies the invoker, which is a remote MIHF. |
| KeyMapping | KEY_MAPPING | This specifies a mapping of a link identifier for which the key is pushed and a lifetime. |

#### 7.4.27.2.3 When generated

This primitive is generated by the local MIHF after receiving an MIH_Push_Key request message from the remote MIHF.

#### 7.4.27.2.4 Effect on receipt

A media specific key is delivered to the corresponding MIH user.

### 7.4.27.3 MIH_Push_key.response

#### 7.4.27.3.1 Function

This primitive is used to indicate that the key installation request has been received and MIH user has executed it.

#### 7.4.27.3.2 Semantics of service primitive

MIH_Push_key.response (
DestinationIdentifier,
LinkTupleIdentifierList,
Status
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| DestinationIdentifier | MIHF_ID | This identifies a remote MIHF that will be the destination of this response. |
| LinkTupleIdentifierList | LIST(LINK_TUPLE_ID) | This identifies a list of links for which keys are pushed. |
| Status | STATUS | This represents the operation result. |

#### 7.4.27.3.3 When generated

This primitive is generated by an MIH user after receiving an MIH_Push_Key.indication primitive.

#### 7.4.27.3.4 Effect on receipt

The local MIHF shall generate an MIH_Push_Key response message to the remote MIHF.

### 7.4.27.4 MIH_Push_Key.confirm

#### 7.4.27.4.1 Function

This primitive is used to notify the MIH user (in MN side) about the status of the requested operation.

#### 7.4.27.4.2 Semantics of service primitive

```
MIH_Push_key.confirm (
                        SourceIdentifier,
                        LinkTupleIdentifierList,
                        Status
                        )
```

Parameters:

| Name | Data type | Description |
|---|---|---|
| SourceIdentifier | MIHF_ID | This identifies the invoker, which is a remote MIHF. |
| LinkTupleIdentifierList | LIST(LINK_TUPLE_ID) | This identifies a list of links for which keys are pushed. |
| Status | STATUS | This represents the operation result. |

#### 7.4.27.4.3 When generated

This primitive is generated after receiving an MIH_Push_Key response message.

#### 7.4.27.4.4 Effect on receipt

A media specific key must be installed in the link layer.

### 7.4.28 MIH_LL_Auth

The primitives defined are to carry out a proactive authentication over MIH between the MN and the PoS using link layer frames. The authentication is conducted with the media specific authenticator that serves the target PoA.

#### 7.4.28.1 MIH_LL_Auth.request

#### 7.4.28.1.1 Function

This primitive carries link-layer frames for authentication purposes.

#### 7.4.28.1.2 Semantics of service primitive

MIH_LL_Auth.request (
DestinationIdentifier,
LinkIdentifier,
LLInformation
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| DestinationIdentifier | MIHF_ID | This identifies a remote MIHF that will be the destination of this request. |
| LinkIdentifier | LINK_TUPLE_ID | This identifies a PoA that is also the authenticator. |
| LLInformation | LL_FRAMES | This carries link layer frames. |

#### 7.4.28.1.3 When generated

This primitive is generated by an MIH user to start an authentication process based on link-layer frames.

#### 7.4.28.1.4 Effect on receipt

The local MIHF shall generate an MIH_LL_Auth request message to the remote MIHF.

### 7.4.28.2 MIH_LL_Auth.indication

#### 7.4.28.2.1 Function

This primitive is used by the remote MIHF to notify the corresponding MIH user about the reception of an MIH_LL_Auth request message.

#### 7.4.28.2.2 Semantics of service primitive

MIH_LL_Auth.indication (
SourceIdentifier,
LinkIdentifier,
LLInformation
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| SourceIdentifier | MIHF_ID | This identifies the invoker, which is a remote MIHF. |
| LinkIdentifier | LINK_TUPLE_ID | This identifies a PoA that is also the authenticator. |
| LLInformation | LL_FRAMES | This carries link layer frames. |

#### 7.4.28.2.3 When generated

This primitive is generated by a remote MIHF after receiving an MIH_LL_Auth request message.

#### 7.4.28.2.4 Effect on receipt

The MIH user must generate an MIH_LL_Auth.response primitive.

### 7.4.28.3 MIH_LL_Auth.response

#### 7.4.28.3.1 Function

This primitive is used by an MIH user to provide the link-layer frames to the local MIHF.

#### 7.4.28.3.2 Semantics of service primitive

MIH_LL_Auth.response (
DestinationIdentifier,
LinkIdentifier,
LLInformation,
Status
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| DestinationIdentifier | MIHF_ID | This identifies a remote MIHF that will be the destination of this response. |
| LinkIdentifier | LINK_TUPLE_ID | This identifies a PoA that is also the authenticator. |
| LLInformation | LL_FRAMES | This carries link layer frames. |
| Status | STATUS | Status of the authentication. |

#### 7.4.28.3.3 When generated

This primitive is generated after receiving an MIH_LL_Auth.indication primitive.

#### 7.4.28.3.4 Effect on receipt

The local MIHF must generate an MIH_LL_Auth response message in order to provide the required information until the authentication is finished.

### 7.4.28.4 MIH_LL_Auth.confirm

#### 7.4.28.4.1 Function

This primitive is used to notify the corresponding MIH user about the reception of an MIH_LL_Auth response message.

#### 7.4.28.4.2 Semantics of service primitive

MIH_LL_Auth.confirm (
SourceIdentifier,
LLInformation,
Status
)

Parameters:

| Name | Data type | Description |
|---|---|---|
| SourceIdentifier | MIHF_ID | This identifies the invoker, which is a remote MIHF. |
| LLInformation | LL_FRAMES | This carries link layer frames. |
| Status | STATUS | Status of the authentication. |

#### 7.4.28.4.3 When generated

This primitive is generated by the remote MIHF after receiving an MIH_LL_Auth response message.

#### 7.4.28.4.4 Effect on receipt

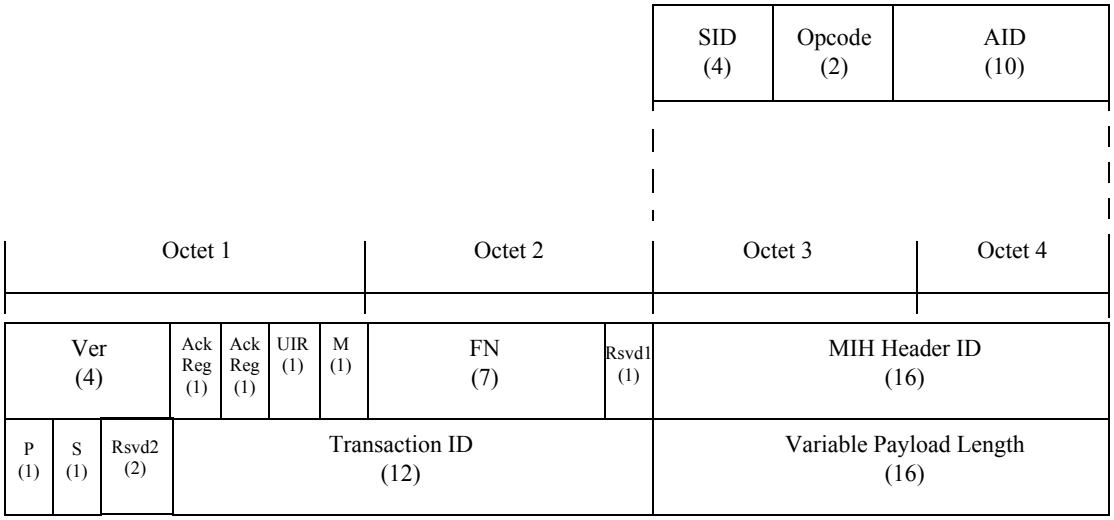
The MIH user may generate an MIH_LL_Auth.request primitive unless the authentication is completed.

# 8. Media independent handover protocol

## 8.4 MIH protocol frame format

### 8.4.1 General frame format

***Replace Figure 28—MIH protocol header format with the following figure*:**




***Insert new rows and change existing row in Table 23 as follows***:

**Table 23—Description of MIH protocol header fields**

| Field name | Size (bits) | Description |
|---|---|---|
| Proactive authentication (P) | 1 | This field is used for indicating that the message is a proactive authentication message. |
| Security association (S) | 1 | This field is used for indicating that a security association exists and the message is protected. |
| Reserved2 | ~~4~~ 2 | This field is intentionally kept reserved. When not used, all the bits of this field are to be set to '0'. |

***Insert the following subclause as shown:***

### 8.4.1a Protected MIH protocol frame format

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


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

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

MIH message can be protected using TLS [RFC5246] or DTLS [RFC4347].

The transport protocol for (D)TLS is the MIH protocol. When the MIH protocol transport is reliable, TLS is used. Otherwise, DTLS is used. The transport protocol entities to be associated with a TLS session are MIHF peers and are identified by MIHF identifiers. The TLS handshake takes place over the MIH protocol and as a result, an MIH SA that contains TLS master key and its child keys, TLS random values and the TLS cipher suite negotiated in the TLS handshake is established between the peers. The detailed description about the protocol interface of using (D)TLS is provided in 9.1.1.

The structure of an MIHS PDU during a TLS handshake is shown in Figure 28b.


**Figure 28b—MIHS PDU during TLS handshake**

Once a (D)TLS handshake is completed, an MIH SA is established, which is determined by the ciphersuite negotiated in the (D)TLS handshake. The structure of protected MIH PDU, when an MIH SA exists, is shown in Figure 28c, where the unprotected MIH service specific TLVs are carried and protected as (D)TLS application data. An MIH header has the S bit set to one. The TLS session ID assigned through TLS handshake is contained in the SAID TLV. The TLS protection can be integrity protected, encrypted, or both. If it is integrity protected, then a message integrity code (MIC) is also included in the security TLV. In this standard, the message integrity code is the same as the message authentication code, for which the acronym MAC is already used for media access control.

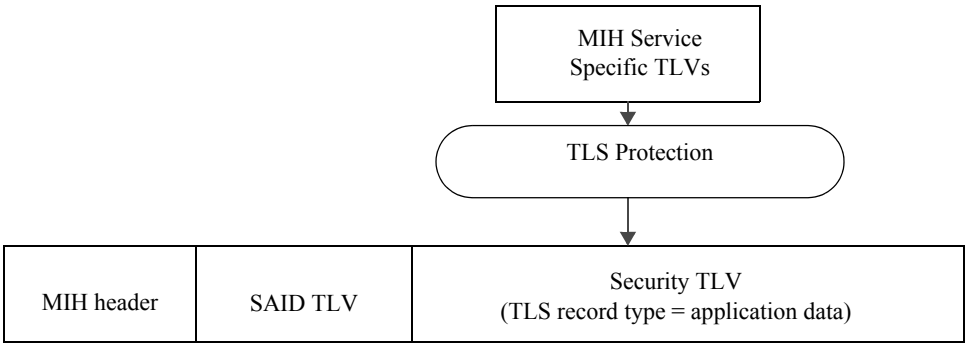



**Figure 28c—MIHS PDU in Existence of MIH SA by TLS**

The MIH message protection procedure is specified in 9.3.

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

An MIH security association (SA) may be established through an MIH service access authentication. An MIH SA is established for a pair of MIHFs. It includes a ciphersuite used for the protection. A security association identifier is assigned by the PoS as a result of successful EAP execution and communicated to the MN via a MIH_Auth request message with a Status indicating Success. Figure 28d shows a protected MIH PDU. The protection procedure is specified in 9.3.1.

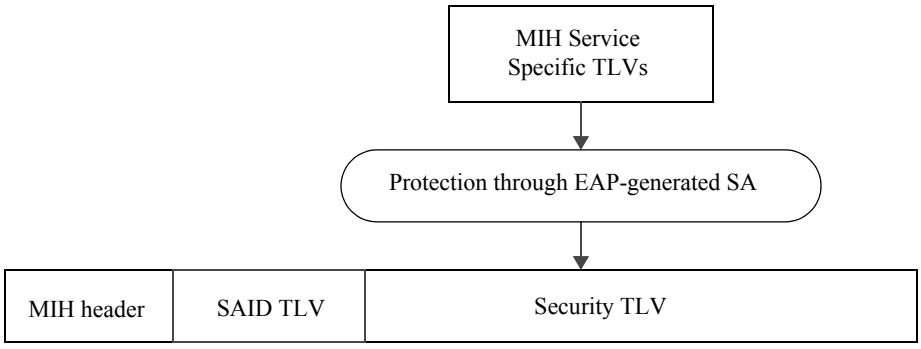



**Figure 28d—MIH PDU Protected by an EAP-generated MIH SA**

### 8.4.1a.3 Protected MIH PDU upon transport address change

If the transport address of an MIHF peer changes over the lifetime of a TLS session or the lifetime of an SA, the mapping between the sender's transport address and the MIHF identifier shall be updated only if the MIHF identifier is the same as that is currently bound to the security association identifier, and an MIH registration request or response message may be contained in the Security TLV. The structure of a protected MIH PDU upon transport address change is shown in Figure 28e.

| MIH header | Source MIHF Identifier TLV | Destination MIHF Identifier TLV | SAID TLV | Security TLV |
|---|---|---|---|---|

**Figure 28e—MIH PDU upon Transport Address Change**

### 8.4.2 Fragmentation and reassembly

#### 8.4.2.1 General

***Change the second paragraph in 8.4.2.1 as follows*:**

An MIH message is fragmented only when MIH message is sent natively over an L2 medium such as Ethernet. The message is fragmented when the message size exceeds aFragmentationThreshold. The size of each of the fragments is the same except the last one, which may be smaller. The maximum fragment size is defined as the maximum value of aFragmentationThreshold, which shall be equal to the Maximum Transmission Unit (MTU) (in octets) of the link layer that is on the path between two MIHF nodes, minus securityOverhead octets, which is the maximum expansion for each protected MIH PDU. When there is no MIH SA, securityOverhead is set to zero. The calculation of securityOverhead when there is an MIH SA is given in Annex K. When the MTU of the link layer between two MIHF nodes is known, the maximum fragment size is set to the MTU (in octets) minus securityOverhead octets. The method of determining such an MTU is outside the scope of this standard. When the MTU of the link layer between two MIHF nodes is unknown, the maximum fragment size is set to the minimum MTU of 1500 octets minus securityOverhead octets. When MIH message is sent using an L3 or higher layer transport, L3 takes care of any fragmentation issue, and the MIH protocol does not handle fragmentation in such cases.

***Insert the following text after Figure 29*:**

When an MIH PDU is protected, the protection is applied to the fragment payload as shown in Figure 29a.

***Insert the following figure*:**

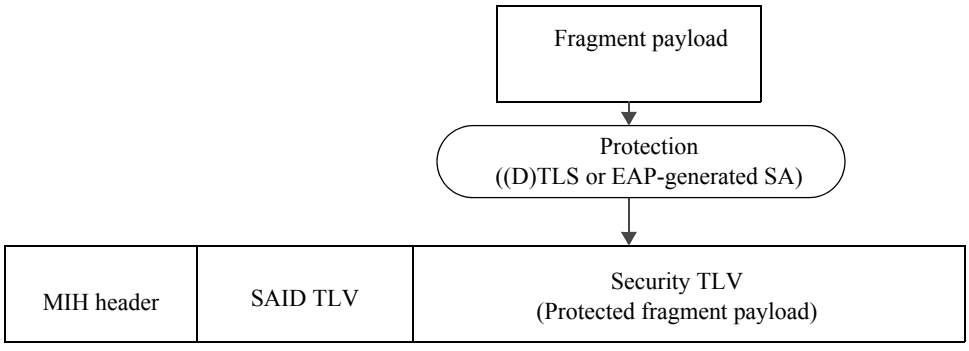



**Figure 29a—Protected fragmented MIH protocol frame format**

### 8.4.2.2 Fragmentation

***Change first paragraph of 8.4.2.2 as follows*:**

When an MIH message is fragmented, the fragmentation is performed within 'Transmit()' procedure in the MIH transaction protocol state machines. In the MIH header, the source MIHF identifier TLV and destination MIHF identifier TLV of the original message are copied to each fragment. When an MIH SA exists, the S bit in the header is set to one and an SAID TLV is included in each fragment. In this case, the source MIHF identifier TLV and destination MIHF identifier TLV of the original message are optional. However the 'variable payload length', 'more fragment', and 'fragment number' fields are updated accordingly for each fragment.

### 8.4.2.3 Reassembly

***Change text as follows*:**

The destination MIHF reassembles the received fragments into an original message. Reassembly is performed outside the MIH transaction state machines. 'MsgIn' and 'MsgInAvail' variables are set only after successful reassembly. An MIHF shall be capable of receiving fragments of arbitrary length.

The following fields are used for reassembling fragments:

- S bit
- MIH message ID
- Transaction ID
- Source MIHF identifier TLV
- Destination MIHF identifier TLV
- SAID TLV (when Source and Destination MIHF identifiers are not present)
- More fragment
- Fragment number

When any fragment of a multi-fragment message has arrived first, the destination MIHF starts a timer referred to as ReassemblyTimer. If this ReassemblyTimer expires before all fragments have been received, the destination MIHF discards those fragments that it has received. A duplicate fragment is discarded.

When S bit is set to one, the fragment is protected. The protected fragment is verified for its integrity at the receiving end. If encryption is applied, it is decrypted to obtain the plaintext fragment. The security association identifier maps the fragment to a pair of source and destination MIHF identifiers that are required for reassembly. The reassembly is performed after all the fragments are verified and decrypted.

An example of an original MIH message and fragmented MIH messages is shown in Annex K.

## 8.6 MIH protocol messages

### 8.6.1 MIH messages for service management

#### 8.6.1.1 MIH_Capability_Discover request

***Insert the following parameters**:*

| **MIH Header Fields (SID=1, Opcode=1, AID=1)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| LinkAddressList (optional)<br>(Link address list TLV) |
| SupportedMihEventList (optional)<br>(MIH event list TLV) |
| SupportedMihCommandList (optional)<br>(MIH command list TLV) |
| SupportedISQueryTypeList (optional)<br>(MIIS query type list TLV) |
| SupportedTransportList (optional)<br>(Transport option list TLV) |
| MBBHandoverSupport (optional)<br>(MBB handover support TLV) |
| SupportedSecurityCapList (optional)<br>(MIH Service Authentication Method list TLV) |

#### 8.6.1.2 MIH_Capability_Discover response

***Insert the following parameters**:*

| **MIH Header Fields (SID=1, Opcode=2, AID=1)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| Status<br>(Status TLV) |
| LinkAddressList (optional)<br>(Link address list TLV) |
| SupportedMihEventList (optional)<br>(MIH event list TLV) |
| SupportedMihCommandList (optional)<br>(MIH command list TLV) |

| |
|---|
| SupportedISQueryTypeList (optional)<br>(MIIS query type list TLV) |
| SupportedTransportList (optional)<br>(Transport option list TLV) |
| MBBHandoverSupport (optional)<br>(MBB handover support TLV) |
| SupportedSecurityCapList (optional)<br>(MIH Service Authentication Method list TLV) |

***Insert 8.6.1.11– 8.6.1.19 after 8.6.1.10 as follows**:*

### 8.6.1.11 MIH_Auth indication

This is used for an MIHF to perform (D)TLS exchange with another MIHF to establish or terminate a (D)TLS-generated MIH SA. It is also used to initiate an MIH service access authentication through EAP or ERP. In the former case, an AuthenticationContent shall be included to carry a TLS record of type handshake, change ciphersuite or alert message. In the latter case, this message is used in two different situations: a) when EAP execution is initiated by the MN; b) when ERP execution is initiated by the PoS. Only in case b), AuthenticationContent shall be included to carry an ERP message (ERP-Initiate/Re-auth-Start). This message shall not be used when EAP execution is initiated by a PoS or when ERP execution is initiated by an MN; an MIH_Auth request message shall be used instead.

| **MIH Header Fields (SID=1, Opcode=3, AID=6)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| AuthenticationContent (optional)<br>(Authentication TLV) |

### 8.6.1.12 MIH_Auth request

This message is used for an MIHF in either an MN or a PoS to send EAP or ERP messages in an MIH service authentication.

| **MIH Header Fields (SID=1, Opcode=1, AID=6)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| Security association ID (optional)<br>(SAID TLV) |
| Nonce (optional)<br>(Nonce TLV) |
| AuthenticationContent (optional)<br>(Authentication TLV) |

| |
|---|
| KeyLifeTime (optional)<br>(Lifetime TLV) |
| Status (optional)<br>(STATUS TLV) |
| CipherSuite(optional)<br>(Ciphersuite TLV) |
| AUTH (optional)<br>(AUTH TLV) |

#### 8.6.1.13 MIH_Auth response

This message is used for an MIHF in either an MN or a PoS to send EAP or ERP messages in an MIH service authentication.

| **MIH Header Fields (SID=1, Opcode=2, AID=6)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| Nonce (optional)<br>(Nonce TLV) |
| AuthenticationContent (optional)<br>(Authentication TLV) |
| KeyLifeTime (optional)<br>(Lifetime TLV) |
| Status (optional)<br>(STATUS TLV) |
| CipherSuite(optional)<br>(Ciphersuite TLV) |
| AUTH (optional)<br>(AUTH TLV) |

#### 8.6.1.14 MIH_Termination_Auth request

This message is used for an MIHF in a PoS to terminate an MIH SA.

| **MIH Header Fields (SID=1, Opcode=1, AID=7)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |

#### 8.6.1.15 MIH_Termination_Auth response

This message is used for an MIHF in an MN to terminate an MIH SA.

| MIH Header Fields (SID=1, Opcode=2, AID=7) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |

#### 8.6.1.16 MIH_Push_key request

This message is used for an MIHF to communicate to another MIHF to push a media specific master session key or media specific master session keys to a specific PoA or PoAs. The corresponding primitive is defined in 7.4.27.1.

| MIH Header Fields (SID=1, Opcode=1, AID=8) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| LinkTupleIdentifierList<br>(Link Tuple Identifier List TLV) |

#### 8.6.1.17 MIH_Push_key response

This message is used for an MIHF to communicate to another MIHF that a media specific master session key or media specific master session keys are installed in a specific PoA or PoAs. The corresponding primitive is defined in 7.4.27.3.

| MIH Header Fields (SID=1, Opcode=2, AID=8) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| LinkTupleIdentifierList<br>(Link Tuple Identifier List TLV) |
| Status (optional)<br>(STATUS TLV) |

#### 8.6.1.18 MIH_LL_Auth request

This message is used for an MIHF to carry link layer frames to conduct an authentication. The corresponding primitive is defined in 7.4.28.1.

| MIH Header Fields (SID=1, Opcode=1, AID=9) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| LinkIdentifier<br>(Link Identifier TLV) |
| LLInformation<br>(Link Layer Information TLV) |

#### 8.6.1.19 MIH_LL_Auth response

This message is used for an MIHF to carry link layer frames to conduct an authentication. The corresponding primitive is defined in 7.4.28.3.

| MIH Header Fields (SID=1, Opcode=2, AID=9) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| LinkIdentifier<br>(Link Identifier TLV) |
| LLInformation<br>(Link Layer Information TLV) |
| Status<br>(Status TLV) |

### 8.6.3 MIH messages for command service

#### 8.6.3.7 MIH_Net_HO_Candidate_Query request

***Insert the following parameter**:*

| MIH Header Fields (SID=3, Opcode=1, AID=4) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| SuggestedNewLinkList<br>(List of link PoA list TLV) |
| SuggestedCandidateAuthenticatorList (the list is empty if no authenticator is suggested.)<br>(Authenticator List TLV) |
| QueryResourceReportFlag<br>(Query resource report flag TLV) |

#### 8.6.3.8 MIH_Net_HO_Candidate_Query response

***Insert the following parameter**:*

| MIH Header Fields (SID=3, Opcode=2, AID=4) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| Status<br>(Status TLV) |
| SourceLinkIdentifier<br>(Link identifier TLV) |
| HandoverStatus (not included if Status does not indicate “Success”)<br>(Handover status TLV) |
| PreferredLinkList (not included if Status does not indicate “Success”)<br>(Preferred link list TLV) |
| PreferredCandidateAuthenticatorList (not included if Status does not indicate “Success”)<br>(Authenticator List TLV) |

#### 8.6.3.9 MIH_MN_HO_Candidate_Query request

***Insert the following parameter**:*

| MIH Header Fields (SID=3, Opcode=1, AID=5) |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| SourceLinkIdentifier<br>(Link identifier TLV) |
| CandidateLinkList<br>(List of link PoA list TLV) |
| QoSResourceRequirements<br>(Handover resource query list TLV) |
| IPConfigurationMethods (optional)<br>(IP address configuration methods TLV) |
| DHCPServerAddress (optional)<br>(DHCP server address TLV) |
| FAAddress (optional)<br>(FA address TLV) |
| AccessRouterAddress (optional)<br>(Access router address TLV) |
| CandidateAuthenticatorList<br>(Authenticator List TLV) |

### 8.6.3.10 MIH_MN_HO_Candidate_Query response

***Insert the following parameter**:*

| **MIH Header Fields (SID=3, Opcode=2, AID=5)** |
|---|
| **Source Identifier** = sending MIHF ID<br>(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID<br>(Destination MIHF ID TLV) |
| Status<br>(Status TLV) |
| SourceLinkIdentifier<br>(Link identifier TLV) |
| PreferredCandidateLinkList (not included if Status does not indicate “Success”)<br>(Preferred link list TLV) |
| PreferredCandidateAuthenticatorList<br>(Authenticator List TLV) |

***Insert Clause 9 and Clause 10 as follows:***

# 9. MIH protocol protection

This clause specifies options and mechanisms to protect remote messages in the media independent handover protocol. The remote messages in the MIH protocol can be protected through the transport protocols at layer 2 or layer 3. The protection through the transport protocols are discussed in Annex O. This clause specifies the mechanisms to protect MIH PDUs at the MIH layer. These mechanisms apply protection to MIH PDUs without depending on transport protocols. They are called MIH specific protection mechanisms. To apply MIH specific protection mechanisms, a mobile node and a point of service (PoS) need to negotiate protection mechanisms and to establish cryptographic keys. MIH message protection shall be accomplished in either of two ways. The first is to use TLS or DTLS and the other is to use EAP or ERP as an MIH service access authentication to establish MIH security associations (SAs). If MIH service access authentication is needed and an authentication server is available, then EAP based authentication and key establishment may be used for establishing an MIH SA. In situations where MIH service access authentication is not required and TLS credentials are available or where MIH service access authentication is required and TLS credentials for access authentication are available at a PoS, then (D)TLS may be used for establishing an MIH SA.

## 9.1 Protection established through MIH (D)TLS

In this option, a mobile node, the client, and a PoS, the server, execute a TLS, specified in IETF RFC 5246, or DTLS, specified in IETF RFC 4347, to establish MIH protection. When the MIH protocol transport is reliable, TLS is used. Otherwise, DTLS is used. In the rest of this standard, (D)TLS is used to denote TLS or DTLS. In a (D)TLS handshake, the mutual authentication is executed through either a pre-shared key or a public key certified by a trusted third party such as a certificate authority. It should be noted that all certificates are required to be validated. The TLS certificate used by the PoS is expected to be provided to the mobile node in a secure manner, e.g., during provisioning process. In this option, the authentication may or may not be related to access control. It can be an access authentication for MIH service if a PoS holds service credentials for the mobile nodes.

After the handshake, a (D)TLS session is established. In this case, the TLS master key and the keys derived from the master key, all the TLS parameters, and TLS ciphersuite negotiated in the TLS handshake form an MIH SA. The (D)TLS security association identifier is carried in each message in the SAID TLV.

In a (D)TLS session, an MIH message is first protected as application data. Then the (D)TLS record is transported by MIH protocol by security TLV.

For a (D)TLS-generated MIH SA, it can be terminated through (D)TLS session termination using an MIH_Auth indication message.

## 9.2 Key establishment through an MIH service access authentication

If MIH service is subscription based and provided by a service provider, then an MIH service access authentication may be needed to authorize the service to a mobile node. In this case, a PoS may obtain a master session key through service access authentication and an MIH security associations can be established through the master session key between the MN and the PoS.

### 9.2.1 MIH service access authentication

In this standard, it is assumed that EAP [IETF RFC3748] or EAP Re-authentication (ERP) IETF RFC5296 is used as the authentication protocol with an MN as the peer and a PoS as the authenticator. An EAP server may be used as a backend server.

For the interface between an MN and a PoS, the MIH protocol is acting as an EAP lower layer. That is, at the MN, an EAP message is generated at the MIHF. When it reaches the PoS, the MIHF in the PoS will process it. For an EAP message from the PoS to the MN, it will also be generated by the MIHF in the PoS. At the MN, the EAP message is passed to the MIHF to process. The protocol stack is illustrated in Figure 30, where it is assumed that an EAP server is employed. After a successful authentication, a master session key (MSK) is exported to the lower layer, that is, MIH layer. An MSK is used to further derive MIH message protection keys.

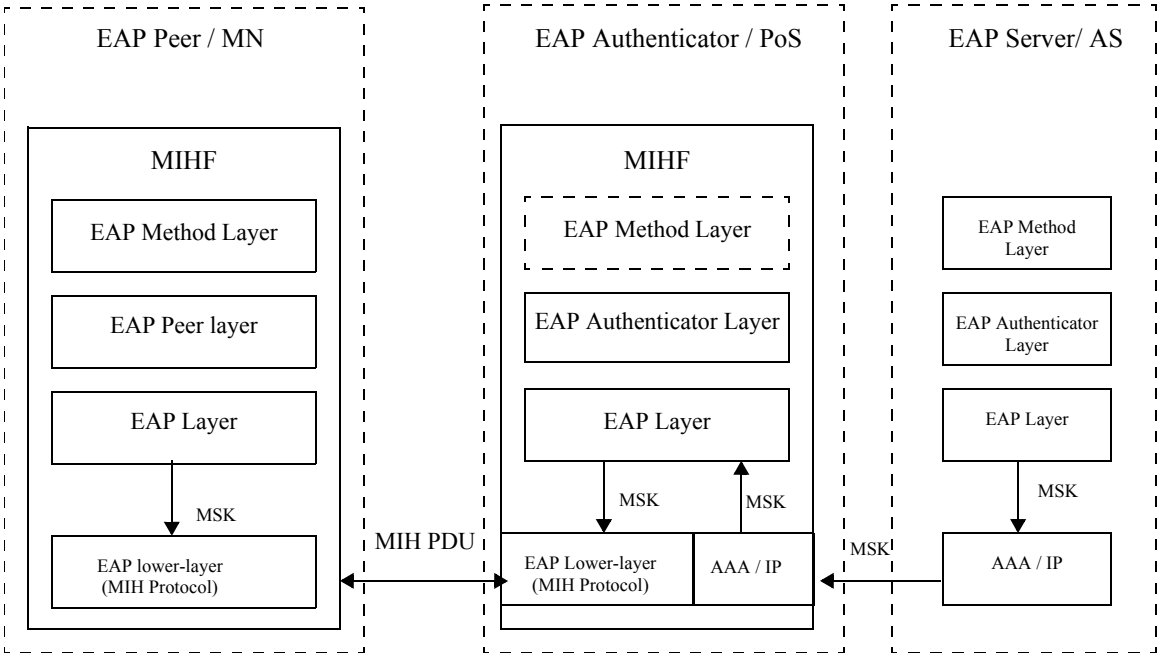



**Figure 30—Protocol Stack of Service Access Authentication (with an EAP Server)**

The authentication is divided into the following phases:

a) *Capability Discovery Phase*. In this phase, both the MN and the PoS exchange unprotected MIH messages for an MN to discover the services that a PoS provides.

b) *MIH Service Access Authentication Phase*. Before starting the MIH access authentication, the MN and the PoS perform a negotiation in order to agree on a ciphersuite and other useful parameters to be used in the authentication and MIH message protection. The negotiation may be initiated either by the MN or by the PoS. Once the negotiation is completed, the MN (acting as the EAP peer) authenticates against the PoS (acting as an EAP authenticator). To achieve this, EAP is transported by MIH protocol between the MN and the PoS. In order to carry out the authentication the PoS may use a backend authentication server (acting as an EAP server) to verify the MN's credentials. In this standard, it is assumed that the EAP methods employed can export keying material (i.e., MSK). Thus, after performing the authentication, keying material (i.e., MSK) will be shared between the MN and the PoS. Specifically, the keying material is exported to MN's and PoS's lower layer (MIH layer) and used to protect the rest of the communication. The message protection mechanisms are specified in 9.3. The protected message format is specified in 8.4. In order to preserve the security of

the exported keying material, the exported MSK is used as a root key to derive session keys which are used to protect the MIH PDUs. The key hierarchy is described in 9.2.2. Note that the authentication procedure could be based on an EAP re-authentication (ERP) in order to perform a fast authentication. In this case, an rMSK is used as the root key to derive the key hierarchy.

c) *Service Access Phase*. At this point, the MN is authenticated and authorized to use the MIH services, agreed and provided by the PoS. The MIH protocol is protected by using the keying material obtained in the MIH Service Access Authentication Phase. This phase is related to 9.2.2 for key derivation and 9.3 for protecting MIH protocol.

d) *Termination phase*. When the MN or the PoS desires to terminate the security association before the security association lifetime expires, either the MN or the PoS can request to terminate.

Figure 31 and Figure 32 illustrate the EAP execution when it is initiated by the MN and when it is initiated by the PoS respectively. In both figures, only the protocol interface between an EAP peer and an EAP authenticator is described. The interface with EAP server is not illustrated. MIH service access authentication messages are defined in 8.6.1.11, 8.6.1.12, and 8.6.1.13. Termination messages are defined in 8.6.1.14 and 8.6.1.15.

Similarly, Figure 33 illustrates an MN initiated ERP execution. Figure 34 and Figure 35 show a PoS initiated ERP execution, where the ERP may be initiated by sending an EAP Request/Identity as shown in Figure 34 or by sending an ERP-Initiate/Re-auth-Start as shown in Figure 35.

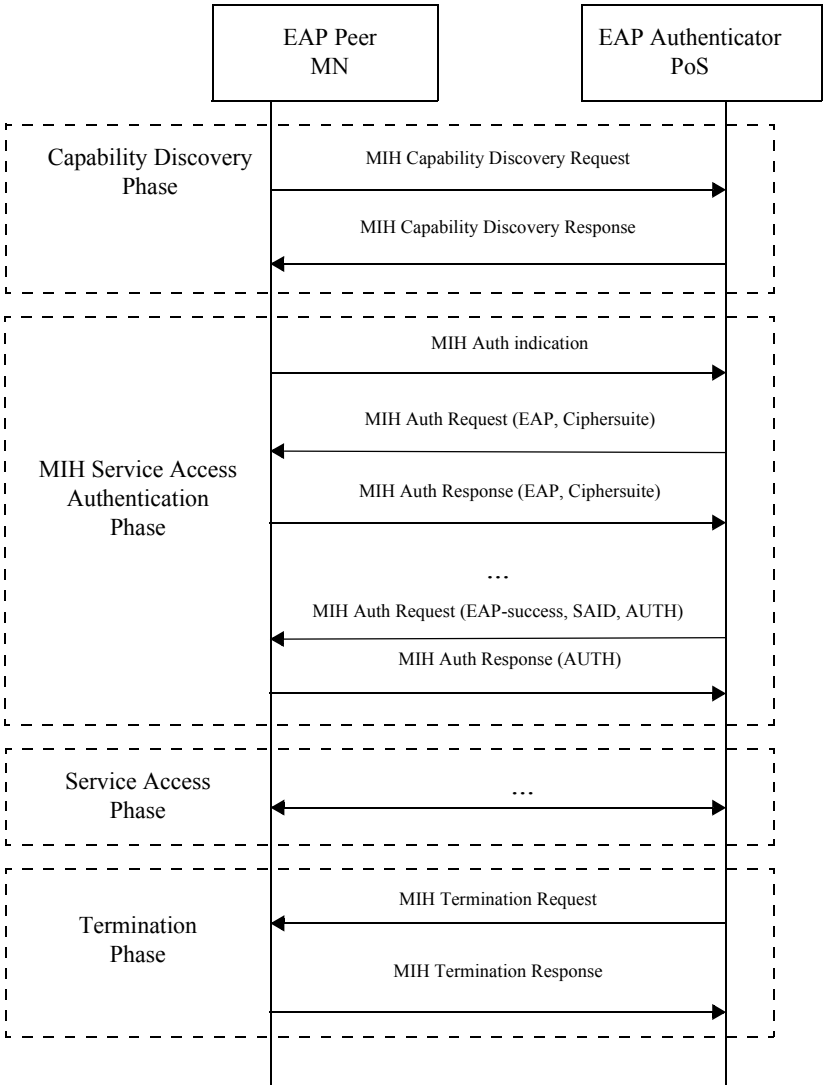



**Figure 31—Main Stages with MN Initiated EAP Execution**

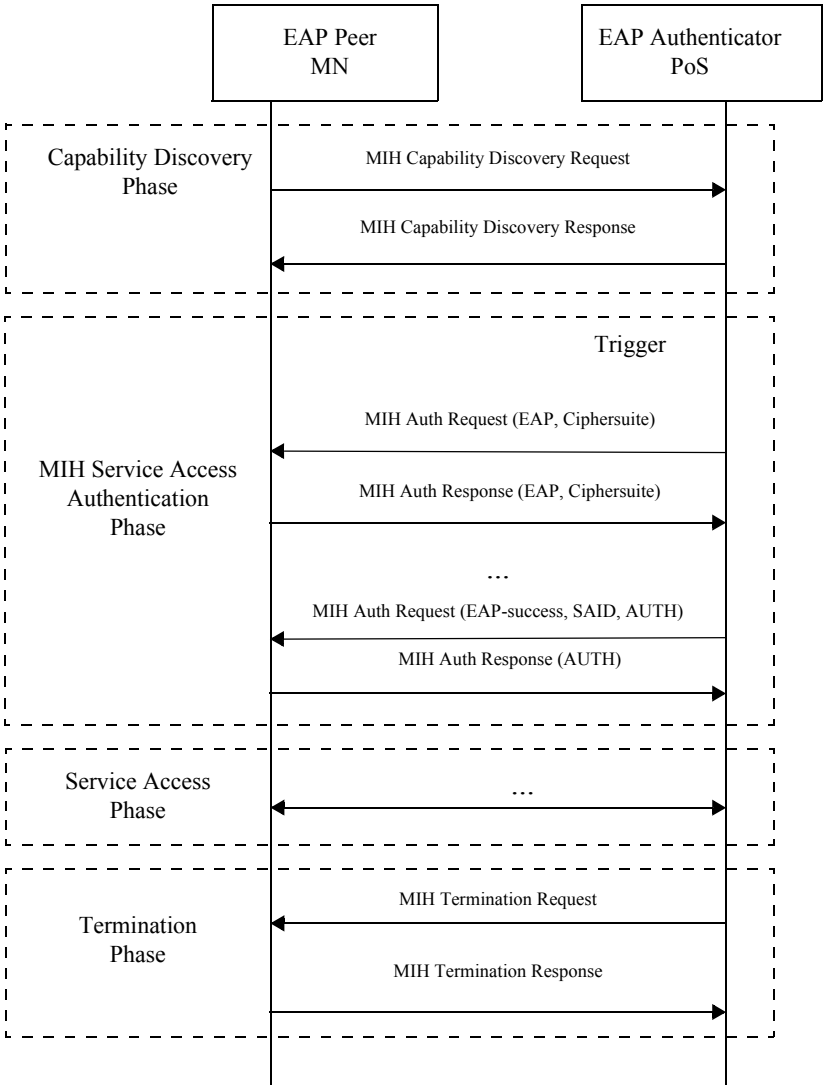



**Figure 32—Main Stages with PoS Initiated EAP Execution**

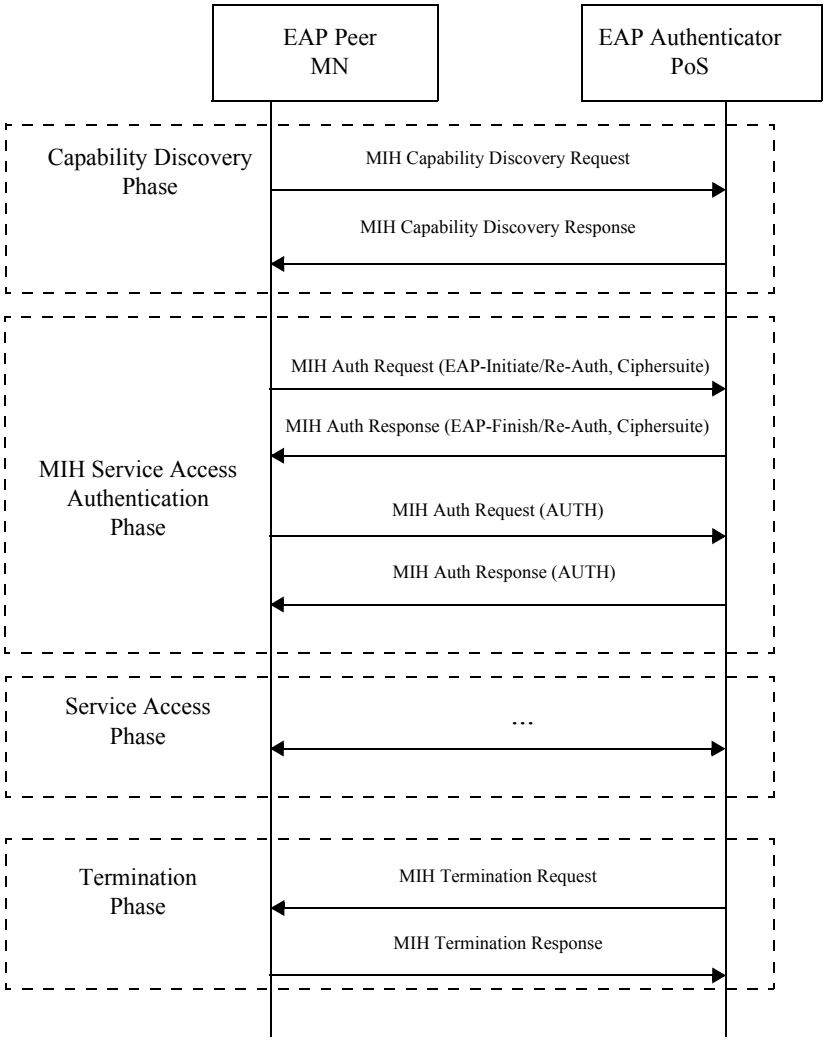



**Figure 33—Main Stages with MN Initiated ERP Execution**

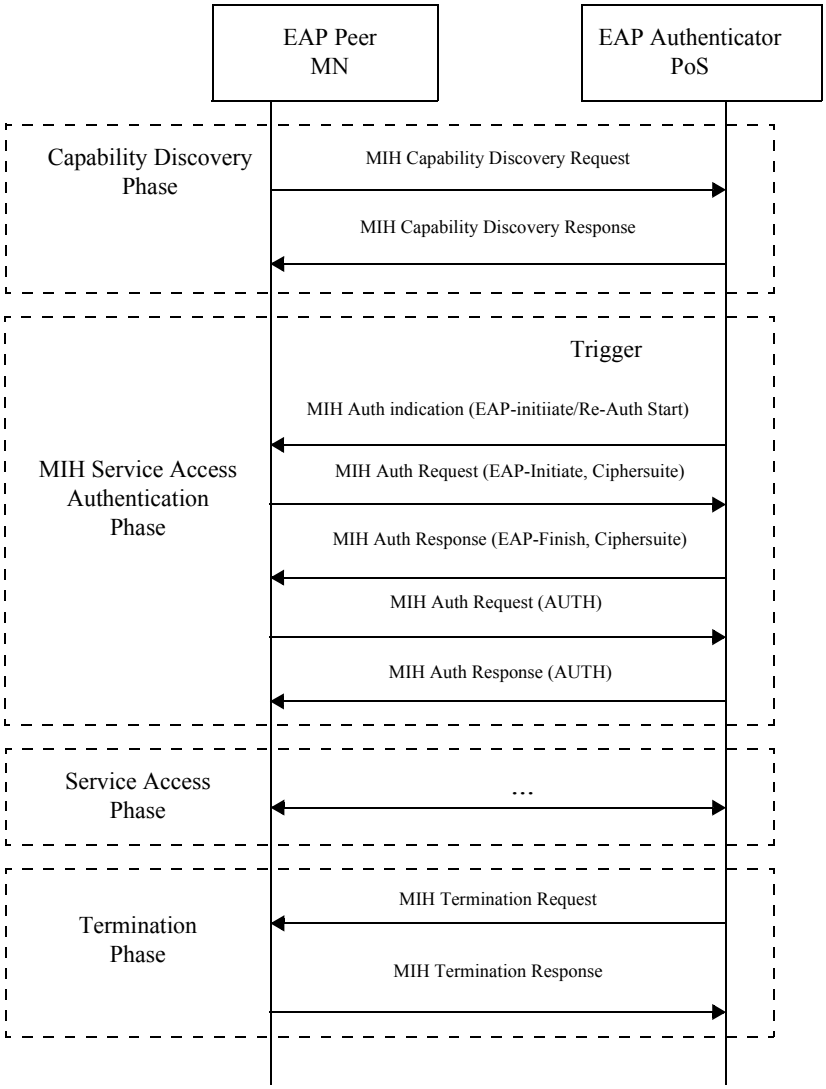



**Figure 34—Main Stages with PoS Initiated ERP Execution (1)**

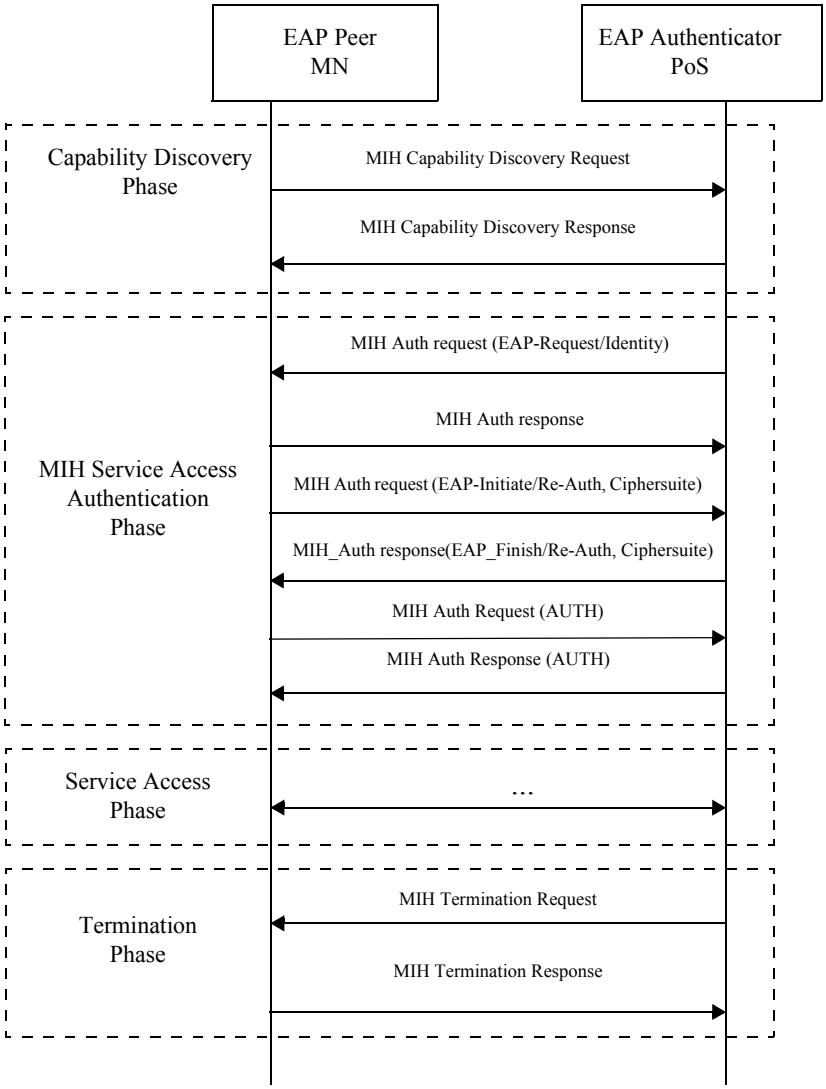



**Figure 35—Main Stages with PoS Initiated ERP Execution (2)**

### 9.2.2 Key derivation and key hierarchy

Upon a successful MIH service access authentication, the authenticator, PoS, obtains a master session key (MSK) or a re-authentication master session key (rMSK). The keys derived from the MSK or rMSK include a 128 bit authentication key (MIAK) used to generate a value AUTH and the session keys determined by the ciphersuite agreed upon between an MN and a PoS. The session keys used for MIH message protection consist of an encryption key (MIEK) only, an integrity key (MIIK) only, or both an encryption key (MIEK) and an integrity key (MIIK). The length, *L,* of the derived keying material, called media independent session key (MISK) are specified in 9.2.3.

For the key derivation, the following notations and parameters are used.

— *K* - key derivation key. It is truncated from a master session key (MSK) or re-authentication MSK (rMSK). The length of *K* is determined by the pseudorandom function (PRF) used for key derivation. If HMAC-SHA-1 or HMAC-SHA-256 is used as a PRF, then the full MSK or rMSK is used as key

derivation key, *K*. If CMAC-AES is used as a PRF, then the first 128 bits of MSK or rMSK are used as key derivation key, *K*.

— *L* - The binary length of derived keying material MISK. *L* is determined by the selected ciphersuite, which is specified in 9.2.3.

— *h* - The output binary length of PRF used in the key derivation. That is, *h* is the length of the block of the keying material derived by one PRF execution. Specifically, for HMAC-SHA-1, *h* = 160 bits; for HMAC-256, *h* =256 bits; for CMAC-AES, *h* = 128 bits.

— *n* - The number of iterations of PRF in order to generate *L*-bits keying material.

— *Nonce-T* and *Nonce-N* - The nonces exchanged during the execution of service access authentication.

— *c* - The ciphersuite code is a one octet string specified for each ciphersuite. The code is defined in 9.2.3.

— *v* - The length of the binary representation of the counter and the length of keying material *L*. The default value for *v* is 32.

— "MISK" - 0x4D495354, ASCII code in hex for string "MISK."

— $[a]_2$ - Binary representation of integer *a* with a given length.

For a given PRF, the key derivation for MISK can be described in the following procedures:

**Fixed input values**: *h* and *v*.

**Input**: *K*, *Nonce-T*, *Nonce-N*, *L*, and ciphersuite code.

**Process**:

a) $n := \lceil L/h \rceil$;

b) If $n > 2^v - 1$, then indicate an error and stop.

c) *Result (0)* := empty string.

d) For $i = 1$ to *n*, do

i) $K(i) := PRF(K, \text{"MISK"} \| [i]_2 \| Nonce\text{-}T \| Nonce\text{-}N \| c \| [L]_2)$.

ii) $Result(i) = Result(i-1) \| K(i)$.

e) Return *Result (n)* and *MISK* is the leftmost *L* bits of *Result (n)*.

Output: *MISK*.

The *MISK* is parsed in such a way that

$$MISK = MIAK \| MIIK \| MIEK. \quad (1)$$

With the above procedure, a key hierarchy is derived as shown in Figure 36.

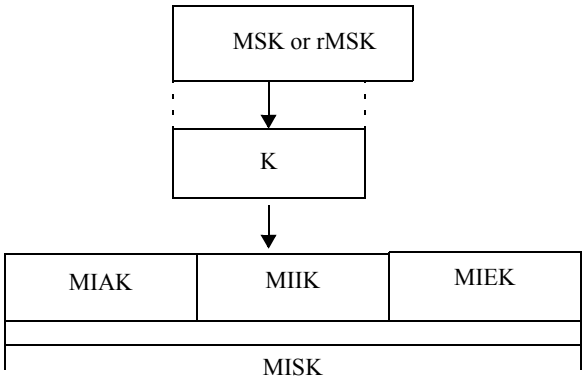



**Figure 36—MIH Key Hierarchy**

### 9.2.3 EAP-generated MIH security association

When an MIH SA is established through an EAP method with key establishment, the SA consists of the keys, the key derivation functions, and the ciphersuite. The key derivation functions, encryption algorithms, and integrity algorithms are specified in Table 24.

**Table 24—Cryptographic algorithms**

| **Encryption algorithm** | **Description** |
|---|---|
| AES_CBC | AES CBC mode ([NIST SP 800-38A]) |
| NULL | No encryption is applied |
| **Integrity algorithm** | **Description** |
| HMAC_SHA1_96 | HMAC-SHA1 with 96 bits MAC ([FIPS 198]) |
| AES_CMAC | AES CMAC mode with 128 bits MAC ([NIST SP 800-38B]) |
| **Authenticated encryption** | **Description** |
| AES_CCM | AES-CCM mode ([NIST SP 800-38C]) |
| **PRF used for key derivation function** | **Description** |
| PRF_CMAC_AES | AES CMAC as PRF in counter mode ([NIST SP 800-108]) |
| PRF_HMAC_SHA1 | HMAC-SHA1 as PRF in counter mode ([NIST SP 800-108]) |
| PRF_HMAC_SHA256 | HMAC-SHA256 as PRF in counter mode ([NIST SP 800-108]) |

The ciphersuites and key lengths are defined in Table 25.

**Table 25—Ciphersuites**

| Code | Encryption algorithm | Integrity algorithm | MISK length (L) |
|---|---|---|---|
| 00000010 | AES_CBC | HMAC_SHA1_96 | 384 |
| 00000100 | NULL | HMAC_SHA1_96 | 256 |
| 00000101 | NULL | AES_CMAC | 256 |
| 00000110 | AES_CCM | | 256 |

A default ciphersuite is defined as AES_CCM. The default PRF is defined as PRF_CMAC_AES. The protection mechanisms for MIH messages are defined in 9.3.

### 9.2.4 Termination

A termination phase is defined as a mechanism to allow either an MN or a PoS to release the resource such as keys, authorized service access, etc. obtained through a service access authentication. Termination shall take place by either of two mechanisms:

a) Termination messages: These messages allow one party to explicitly inform another party the current authentication status is terminated. This option is supported by MIH_Termination_Auth messages defined in 8.6.1.14 and 8.6.1.15.

b) State timeout: A lifetime is defined for an MIH SA. After the time period defined by the lifetime, the MIH SA is terminated. The lifetime of the SA must be no longer than the MSK or rMSK lifetime, and communicated to the MIH node acting as the EAP peer by the Lifetime TLV included in MIH_Auth request and MIH_Auth response messages defined in 8.6.1.12 and 8.6.1.13.

## 9.3 MIH message protection mechanisms for EAP-generated SAs

### 9.3.1 MIH_Auth message protection

The MIH_Auth messages are not protected using the MIH SA established after a successful media independent service access authentication. MIH_Auth messages are integrity protected by including an AUTH TLV generated using MIAK derived from the MSK or rMSK, as described in 9.2.2, with a PRF. The AUTH TLV value is generated as follows:

*AUTH TLV value = PRF(K, "AUTH-TLV" | MIH_Auth message| MNCiphersuite | PoSCiphersuite),*

where

— K -MIAK

— “AUTH-TLV”- 0x415554482D544C56, ASCII code in hex for string “AUTH-TLV”

— MIH_Auth message - an MIH_Auth message in which AUTH TLV filled with 0s

— MNCiphersuite - Ciphersuite TLV sent by the MN

— PoSCiphersuite - Ciphersuite TLV sent by the PoS

PRF function is one of the following as negotiated:

a) PRF_CMAC_AES

b) PRF_HMAC_SHA1

c) PRF_HMAC_SHA256

The PRF output length must be truncated to 128 bits. If the PRF output length is more than 128 bits, the 128 leftmost bits of the output must be used as the AUTH TLV value.

### 9.3.2 MIH PDU protection procedure

Depending on the selected ciphersuite, an MIH PDU may be encrypted, integrity protected, or protected in both aspects. Correspondingly, an SA may identify an encryption key (MIEK), an integrity key (MIIK), or both MIEK and MIIK. When both encryption and integrity protection are applied, they may be accomplished by two algorithms such as AES in CBC mode and HMAC-SHA1_96 or by one authenticated encryption algorithm such as AES in CCM mode.

The portion to be protected in an MIH message includes the MIH service specific TLVs. That is, the source MIHF identifier TLV and the destination MIHF identifier TLV are not protected. The protection is applied based on an SA. An SAID TLV is carried in place of source and destination MIHF identifier TLVs except for the case of transport address change where both an SAID TLV and source and destination MIHF identifier TLVs are carried as described in 8.4.1a. An example procedure is illustrated in Figure 37.

When fragmentation is applied to an MIH PDU, then instead of service specific TLVs, the data to be protected comprise a fragment payload. The values in the header fields M (more fragment) and FN (fragment number) are the same after the protection.

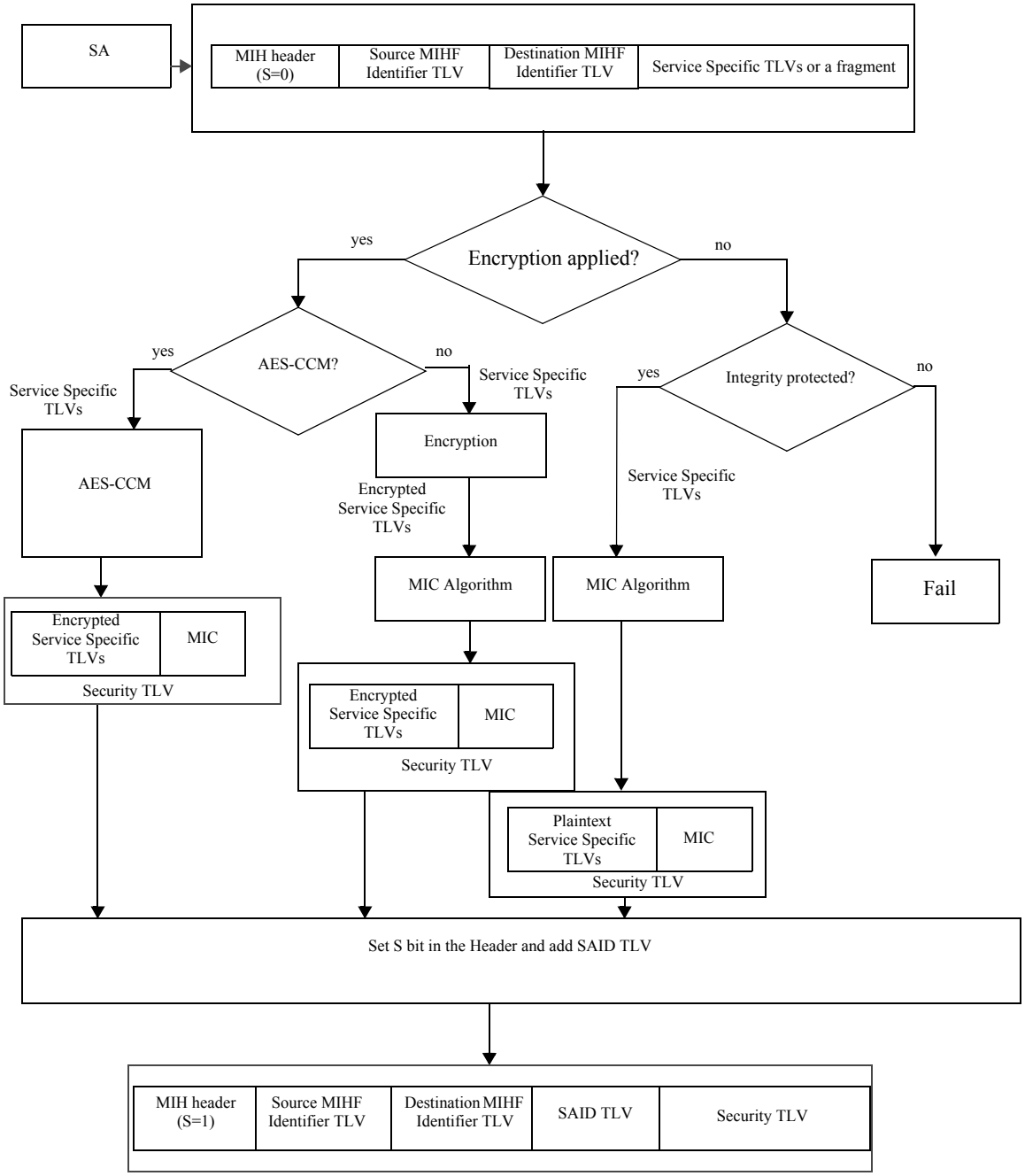



**Figure 37—MIH PDU protection procedure**

### 9.3.3 MIH PDU protection by AES-CCM

AES in CCM mode as specified in NIST SP 800-38C shall be the default ciphersuite. The parameters used in AES-CCM, the nonce construction, the operational procedures, and the security TLV under AES-CCM protection shall be set according to the rules given in 9.3.3.1 through 9.3.3.3.

#### 9.3.3.1 AES-CCM Parameters

For AES-CCM the following parameter values shall be set:

a) *t* - The length of MIC is 12 octets (96 bits).

b) *n* - The length of the nonce *N* is 13 octets (104 bits).

c) *q* - The length of the binary representation of the octet length of the data to be encrypted is 2 octets (16 bits).

#### 9.3.3.2 Construct AES-CCM Nonce

AES-CCM uses a nonce to construct an initialization vector and also the counter. CCM requires a unique nonce value for each MIH message protected by a given MIEK. In this standard, the nonce is 13 octets and consists of the following three portions.

a) Transaction ID (12 bits, from the MIH header) plus 4 reserved bits (set to zero);

b) Sequence number (10 octets, denoted as *SN0*, *SN1*, ..., *SN9*) starting from all zeros; and

c) FN (7 bits, from the MIH header) plus 1 reserved bit (set to zero).

The nonce construction is illustrated in Figure 38.

| Transaction ID (12) | Resv (4) | *SN0*, *SN1* ..., *SN9* (80) | FN (7) | Resv (1) |
|---|---|---|---|---|

**Figure 38—AES-CCM Nonce Construction**

The *SN* is increased by a positive number for each MIH PDU. The SN shall never repeat for a series of encrypted MIH PDUs using the same MIEK. For a given SA, each of MIHFs keeps an SN, which is the highest as used for a given MIEK.

#### 9.3.3.3 Operational procedures in AES-CCM

##### 9.3.3.3.1 Encapsulation

For a given SA, the prerequisites for AES-CCM encapsulation includes an encryption key MIEK, an AES block cipher encryption block, and the values of parameters *t*, *n*, and *q*. The plaintext, *P,* to be encrypted and authenticated is formed by concatenating all the service specific TLVs as presented in MIH PDU with the padding. In this standard, the associated data, A, is null. The data, *P,* is partitioned with necessary padding to 16-octet blocks *B1*, *B2*, ..., *Br* as specified in SP 800-38C. The octet block, *B0,* is an initialization vector and formed with 1-octet flags, 13-octet nonce *N*, and 2-octet integer *Q*, where *Q* is the octet length of *P*. The format of *B0* is illustrated in Figure 39

| Flags (1 octet) | Nonce (13 octets) | *Q* (2 octets) |
|---|---|---|

**Figure 39—Format of *B0***

The flags are formed by the following data:

— 1 reserved bit, which is set to zero;

— 1 bit flag for the associated data, which is zero;

— 3 bits to represent ($t$-2)/2, which is 101 ($t$ =12);

— 3 bits to represent $q$-1, which is 001 ($q$ = 2).

The counter *Ctr(i)*, $i$ = 0, 1, ..., r, is formed with 1-octet flags; 13-octet nonce $N$; and 2-octet integer $i$. The format of *Ctr(i)* is illustrated in Figure 40

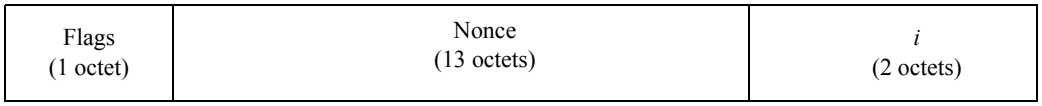



**Figure 40—Format of Counter *Ctr(i)***

The flags for *Ctr(i)* is 00000001.

The encapsulation of an MIH PDU consists of the following steps:

a) Fetch Transaction ID and FN from the MIH header.

b) Increment a positive number of *SN* to update the SN.

c) Construct the nonce, $N$, as described in 9.3.3.2.

d) Input $N$ and $P$ to AES-CCM generation-encryption process as specified in SP 800-38C. The *B0* and all the counter numbers are formed as described in Figure 39 and Figure 40, respectively.

e) Obtain the output, C, of AES-CCM.

#### 9.3.3.3.2 Decapsulation

For a given SA, the prerequisites for AES-CCM decapsulation includes an encryption key MIEK, AES block cipher encryption block, the parameters $t$, $n$, and $q$.

The decapsulation of a protected MIH PDU consists of the following steps.

a) Fetch Transaction ID and FN from the MIH header.

b) Fetch *SN* from the security TLV.

c) Construct the nonce, $N$, as described in 9.3.3.2.

d) Input $N$ and $C$ to AES-CCM decryption-verification process as specified in SP 800-38C. The *B0* and all the counter numbers are formed as described in 9.3.3.3.1.

e) Obtain the output, $P$, or “INVALID”.

### 9.3.3.4 Format of security TLV

The ENCR_BLOCK data of the Security TLV in a protected MIH message with AES-CCM is formed by SN and ciphertext C, which is the ciphertext of P and T (MIC). It is illustrated in Figure 41. Since MIC is carried in the ENCR_BLOCK data, the INTEGR_BLOCK in MIH_SPS_RECORD is not chosen for AES-CCM.

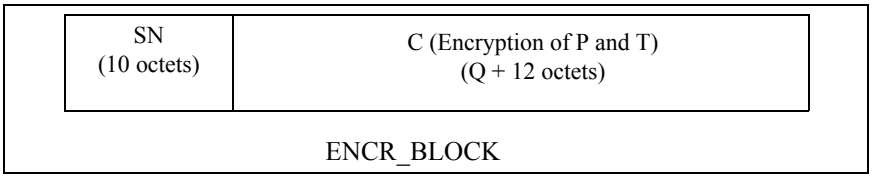



**Figure 41—Security TLV for AES-CCM**

### 9.3.4 MIH PDU protection by AES in CBC mode and HMAC-SHA1-96

This ciphersuite includes two algorithms, encryption algorithm AES CBC and message authentication code algorithm HMAC-SHA1-96. When an MIH PDU is protected, encryption is applied first and an MIC is generated on the ciphertext. The MIC is 12 octets (96 bits). In order to use the ciphersuite, two keys (MIIK and MIEK) are used for encryption/decryption and generation/verification of MIC respectively.

#### 9.3.4.1 Initialization vector for AES in CBC mode

Encryption using AES in CBC mode needs an initialization vector, IV, of 16 octets (128 bits), *IV* = (*IV0*, *IV1*, ..., *IV15*). It can be selected randomly when encryption is executed. It is also needed for decryption. Therefore, for each protected MIH PDU, an IV is included in ENCR_BLOCK as a part of security TLV.

#### 9.3.4.2 Operational procedures in applying AES CBC and HMAC-SHA1-96

##### 9.3.4.2.1 Encapsulation

The encapsulation of an MIH PDU includes the following steps:

a) Select a 16-octet initialization vector, *IV*.

b) Pad the plaintext, *P,* to a length of a multiple of 16 octets (128 bits) so that the padded plaintext can be represented as in *n* blocks *P0*, *P1*, .., *Pn-1*, each of which is 16 octets.

c) Apply AES CBC mode with *IV* and key, MIEK, on *P0*, *P1*, ..., *Pn-1* to obtain ciphertext *C0*, *C1*, ..., *Cn-1*.

d) Input *M = IV||C0||C1||...||Cn-1*, where "||" means concatenating, as the message and MIIK as the key to HMAC-SHA1. Here padding may be needed to make the input message length to be a multiple 64 octets (512 bits). The most significant 12 octets of the output of HMAC-SHA1 is the MIC.

e) Output *C0*, *C1*, ..., *Cn-1* and MIC.

##### 9.3.4.2.2 Decapsulation

The decapsulation of a protected MIH PDU includes the following steps:

a) Fetch the ciphertext *C0*, *C1*, ..., *Cn-1* and the initialization vector, *IV*, from ENCR_BLOCK of security TLV.

b) Fetch the MIC from the INTG_BLOCK of security TLV.

c) Input *M = IV||C0||C1||...||Cn-1*, where "||" means concatenating, as the message and MIIK as the key to HMAC-SHA1. Here padding may be needed to make the input message length a multiple 64 octets (512 bits). Compare the most significant 12 octets of the output of HMAC-SHA1 with the MIC. If they are identical, go to the next step. Otherwise, output "INVALID".

d) Input ciphertext, *C0*, *C1*, ..., *Cn-1*, and MIEK to AES CBC mode to obtain plaintext, *P0*, *P1*, ..., *Pn-1*.

e) Remove the padding if it is applied to obtain the plaintext, *P*.

f) Output *P*.

#### 9.3.4.3 Format of security TLV

When an MIH PDU is protected by AES in CBC mode and HMAC-SHA1-96, both ENCR_BLOCK and INTG_BLOCK appear in the security TLV. The initialization vector IV and the ciphertext, *C0*, *C1*, ..., *Cn-1*, are included in ENCR_BLOCK and MIC is in INTG_BLOCK as shown in Figure 42.

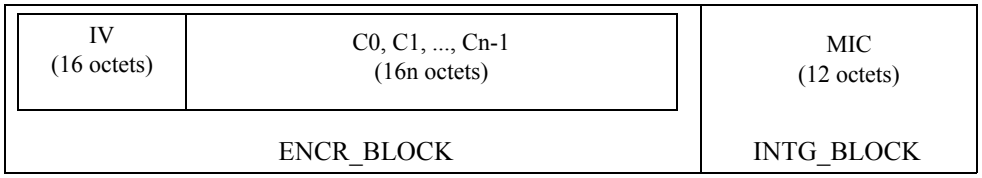



**Figure 42—Security TLV for AES CBC and HMAC-SHA1-96**

### 9.3.5 MIH PDU protection by HMAC-SHA1-96

This ciphersuite includes one message authentication code algorithm, HMAC-SHA1-96. It generates a 12 octets (96 bits) MIC over the protected data using key MIIK.

#### 9.3.5.1 MIC generation and verification

##### 9.3.5.1.1 MIC generation

A MIC is generated in the following steps:

a) The data, *P*, to be protected is padded to a length of a multiple of 64 octets (512 bits).

b) Input the padded data and the key, MIIK, to HMAC-SHA1.

c) Obtain output of HMAC-SHA1.

d) Truncate the output of HMAC-SHA1 to obtain the most significant 96 bits as the MIC.

e) Output MIC.

##### 9.3.5.1.2 MIC verification

A MIC is verified in the following steps:

a) Fetch the data, *P*, from the ENCR_BLOCK of security TLV. Pad it to a length of a multiple of 64 octets (512 bits).

b) Fetch the MIC from INTG_BLOCK of security TLV.

c) Input the padded data and the key, MIIK, to HMAC-SHA1.

d) Obtain output of HMAC_SHA1.

e) Compare the most significant 96 bits of the output of HMAC-SHA1 with MIC.

f) If they are identical, output “VALID”; Otherwise, output”INVALID”.

#### 9.3.5.2 Format of security TLV

When an MIH PDU is protected by HMAC-SHA1-96, the plaintext is included in ENCR_BLOCK, even though it is not encrypted and the MIC is in the INTG_BLOCK as shown in Figure 43.

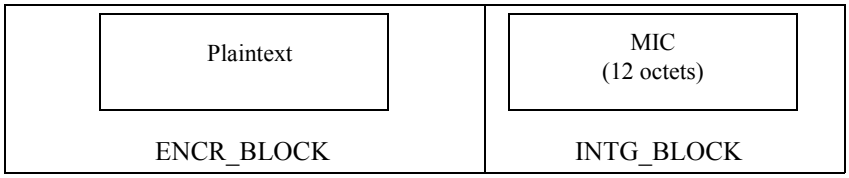



**Figure 43—Security TLV for HMAC-SHA1-96**

### 9.3.6 MIH PDU protection by AES-CMAC

This ciphersuite includes one MAC algorithm, AES-CMAC. It generates a 12 octets (96 bits) MIC over the protected data using key, MIIK.

#### 9.3.6.1 MIC generation and verification

##### 9.3.6.1.1 MIC generation

A MIC is generated in the following steps:

a) Input the data, *P*, to be protected and the key, MIIK, to AES-CMAC. (For AES-CMAC, the padding is specified as a part of the algorithm.)

b) Obtain output of AES-CMAC.

c) Truncate the 16 octets (128 bits) output of AES CMAC to obtain the most significant 96 bits as the MIC.

d) Output MIC.

##### 9.3.6.1.2 MIC verification

A MIC is verified in the following steps:

a) Fetch the data, *P*, from the ENCR_BLOCK of security TLV.

b) Fetch the MIC from INTG_BLOCK of security TLV.

c) Input the data, *P*, to be protected and the key, MIIK, to AES-CMAC. (For AES-CMAC, the padding is specified as a part of the algorithm.)

d) Obtain output of AES-CMAC.

e) Compare the most significant 96 bits with the MIC.

f) If they are identical, output “VALID”; Otherwise, output “INVALID”.

#### 9.3.6.2 Format of security TLV

When an MIH PDU is protected by AES-CMAC, the plaintext is included in the ENCR_BLOCK, even though it is not encrypted and the MIC is in the INTG_BLOCK as shown in Figure 44.

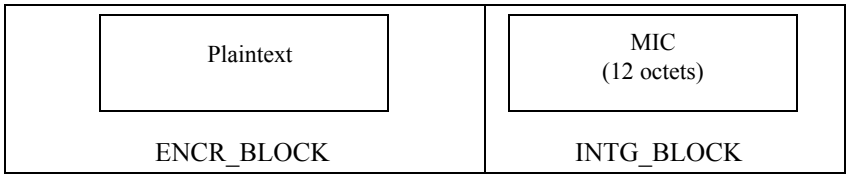



**Figure 44—Security TLV for AES-CMAC**

## 9.4 Common procedures

The following procedures are common for both (D)TLS- and EAP-generated MIH SAs.

### 9.4.1 Sending

For sending a remote MIH message in a protected manner, an MIH PDU is created in the following steps:

a) At the sender, which can be an MN or a PoS, the MIH user generates an MIH primitive and passes it to the MIHF.

b) The MIHF at the sender constructs an MIH PDU. If an MIH security association (SA) exists, then the MIHF at the sender applies (D)TLS protection algorithms specified by the negotiated ciphersuite in the handshake to the MIH PDU and then encapsulates the protected MIH PDU in a security TLV. If no MIH SA exists, then the MIH PDU is passed to the transport protocol of the MIH message.

c) The security TLV is encapsulated in an MIH PDU with the S bit in the MIH header set to one.

d) The MIHS PDU is passed to the transport protocol of the MIH message.

### 9.4.2 Receiving

For receiving a protected MIH message from a remote entity, the protected MIH PDU is processed in the following steps:

a) At the receiver, which can be an MN or a PoS, the MIHF receives a protected MIH PDU from the transport protocol of the MIH message.

b) If the S bit is set to one in the header, the MIHF processes security TLV and extract the plaintext MIH PDU. Otherwise, it takes MIH PDU as is.

c) The MIHF creates an MIH primitive from MIH PDU and passes it to the MIH user at the receiver.

The processing steps at the sender and receiver are described in Figure 45.

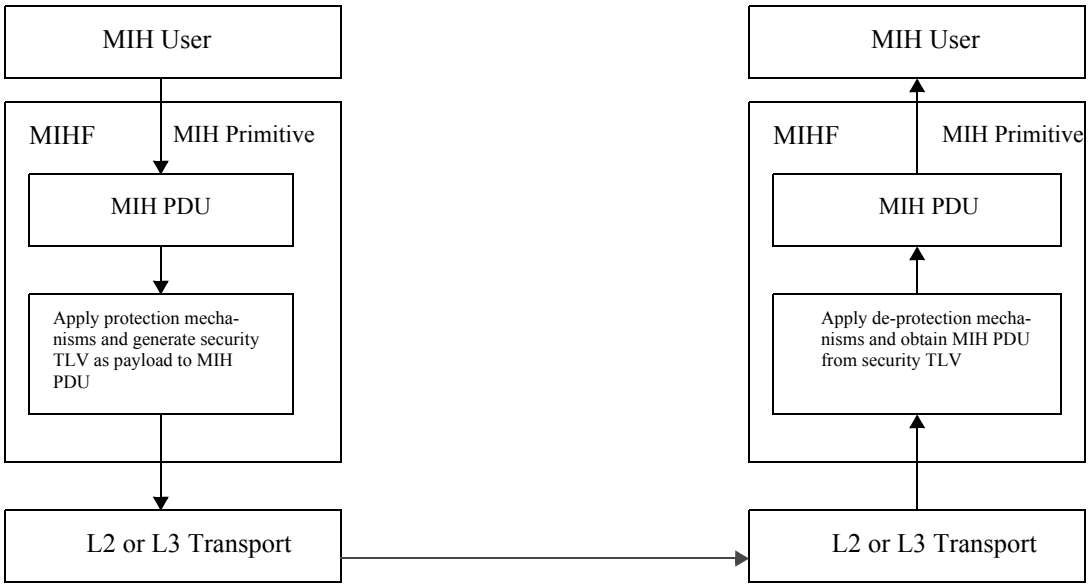



**Figure 45—Sending and Receiving Protected MIH PDU**

The transport protocol entities to be associated with an MIH SA are MIHF peers and are identified by MIHF identifiers. Therefore, the transport address of an MIHF can change over the lifetime of an MIH SA as long as the mapping between the transport address and the MIHF identifier of an MIHF is maintained.

# 10. Proactive authentication

In a handover from a service PoA to a target PoA, a mobile node may need to authenticate to the target network through an authentication mechanism required by the target network. This clause specifies the mechanisms to use MIH to assist proactive authentications to reduce the latency due to media access authentication and key establishment.

This standard introduces two options to conduct the proactive authentication with a targeted network. The first option is called unbundled media access proactive authentication. In such a proactive authentication, an MN conducts an authentication with the targeted network as it is required for accessing that network through a specific media. In this case, the authenticator is a media specific authenticator (MSA). The authentication messages are passed between the MN and the MSA through an MIH PoS. The authentication messages between the MN and the PoS are carried through MIH messages. The second option is to bundle the media access proactive authentication to the MIH service access authentication. In this case, at the end of MIH service access authentication, the MN and the PoS also establish a key(s) for a target PoA(s). The key(s) are distributed to the PoA(s) so that when a handover to one of the PoAs happens, the MN can establish a protected link with the PoA. The MIH message exchange between an MN and a PoS is common to both bundled and unbundled proactive authentication. The only difference is that the bundled proactive authentication uses a key established through MIH service access authentication. The MIH message exchange for bundled and unbundled proactive authentication is described in 10.1.

## 10.1 Media specific proactive authentication

In a media access proactive authentication, a PoS passes authentication messages between the mobile node and a media specific authenticator (MSA). The protocol stacks in each interface are illustrated in Figure 46. In scenarios where MSA/Target PoA is reachable via same media as MN and PoS, EAP messages received at PoS are directly forwarded to the target PoA.

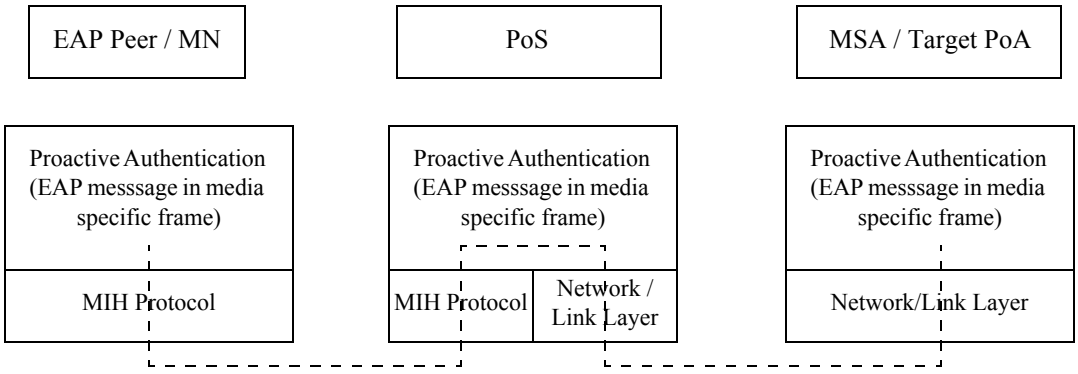



**Figure 46—Protocol Stack for MIH Supported Proactive Authentication**

#### 10.1.1 Procedures in a media specific proactive authentication

An MIH assisted media specific proactive authentication includes the following main procedures.

##### 10.1.1.1 PoS and candidate media specific authenticator discovery

Before an MN initiates an MIH assisted proactive authentication, the MN needs to know the PoS's address and the candidate media specific authenticators' link layer addresses. The corresponding candidate MSAs' addresses can be discovered by using the information elements (IEs) specified in 6.5.4.

##### 10.1.1.2 Proactive authentication through EAP or ERP

In order to execute a proactive authentication, the EAP or ERP messages are encapsulated in the extended MIH messages as L2 frames. When the PoS receives an encapsulated EAP or ERP message, it decapsulates it, then forwards it to the candidate media specific authenticator (MSA). The EAP or ERP messages are encoded as OCTET_STRING.

##### 10.1.1.3 Media specific association handshake

When the MN decides to handover to a candidate network, the MN and the PoA, which is associated with the MSA, perform a media specific association based on the keying material derived by the proactive authentication. For example, the media specific handshake could be a 4-way handshake as in an IEEE 802.11 network. A media specific handshake may further derive Media specific session keys to protect the communication between the MN and the PoA once it is attached to it.

#### 10.1.2 Proactive authentication message format

When a proactive authentication is executed through EAP [RFC3748] or ERP [RFC5296], the EAP packets are carried by MIH messages. MIH primitives for the link layer frames are defined in 7.4.28 for proactive authentication. The messages are defined in 8.6.1.18 and 8.6.1.19. The MIH messages for proactive authentication shall be protected by an MIH SA.

### 10.2 Bundling media access authentication with MIH service access authentication

When the trust relationship between media specific network access provider and the MIH service provider allows, a proactive authentication can be optimized by bundling the media access authentication with an MIH service access authentication. In this case, at the end of a successful service access authentication, a PoS will derive not only keys for MIH message protection as defined in 9.2.2 but also a key called media specific root key (MSRK). This key will be further used to derive a key or keys called media specific pairwise master keys (MSPMKs) to be used by a target PoA or PoAs.

#### 10.2.1 Media specific key derivation

##### 10.2.1.1 Derivation of media specific root key (MSRK)

After a successful service access authentication through EAP or ERP, a master session key (MSK) or a re-authentication MSK (rMSK) is generated in the MN and the PoS. The media specific root key (MSRK) is derived from MSK or rMSK.

For the media specific root key derivation, the following notations and parameters are used:

— *K* - key derivation key. It is truncated from a master session key (MSK) or re-authentication MSK (rMSK). The length of *K* is determined by the pseudorandom function (PRF) used for key derivation. If HMAC-SHA-1 or HMAC-SHA-256 is used as a PRF, then the full MSK or rMSK is used as key

derivation key, *K*. If CMAC-AES is used as a PRF, then the first 128 bit of MSK or rMSK is used as key derivation key, *K*.

— *h* - The output binary length of PRF used in the key derivation. That is *h* is the length of the block of the keying material derived by one PRF execution. Specifically, for HMAC-SHA-1, *h* = 160 bits; for HMAC-256, *h* =256 bits; for CMAC-AES, *h* = 128 bits.

— *Nonce-T* and *Nonce-N* - The nonces exchanged during the execution of service access authentication.

— "MSRK" - 0x4D53524B, ASCII code in hex for string "MSRK."

The MSRK derivation is described as follows:

**Input**: *K*, *Nonce-T*, *Nonce-N*.

**Process**:

a) *MSRK:= PRF(K, "MSRK" || Nonce-T || Nonce-N)*.

b) Return *MSRK*.

**Output**: *MSRK*

The binary length of *MSRK* is *h*. Depending on the PRF used for the MSRK derivation, it can be 128 bits, 160 bits, or 256 bits. The MSRK is used to derive media specific pairwise master keys (MSPMK).

#### 10.2.1.2 Derivation of media specific pairwise master keys (MSPMKs)

Each MSPMK is derived specifically for a PoA. For the media specific pairwise master key (MSPMK) derivation, the following notations and parameters are used:

— *K*- key derivation key. It can be a full length of *MSRK* or a portion of *MSRK*. Specifically, the length of *MSRK* is *h* which is determined by the PRF used for key derivation. If in MSRK derivation and in MSPMK derivation, the same PRF is used, the MSPMK derivation will be able to use the full length *MSRK*. However, in case that HMAC-SHA1 or HMAC-SHA256 is used in MSRK derivation, but CMAC-AES is used in MSPMK derivation, then only the first 128 bits of MSRK is used as a key derivation key in the MSPMK derivation.

— *MN_LINK_ID* - A link layer identity of the mobile node.

— *PoA_LINK_ID* - A link layer identity of a target point of attachment (PoA).

— "*MSPMK*" - 0x4D53504D4B, ASCII code in hex for string "MSPMK".

The MSPMK derivation is described as follows:

**Input**: *K*, *MN_LINK_ID*, *PoA_LINK_ID*.

**Process**:

a) *MSPMK:= PRF(K, "MS-PMK" || MN_LINK_ID|| PoA_LINK_ID)*.

b) Return *MSPMK*.

**Output**: *MSPMK*.

The binary length of *MSPMK* is *h*. Depending on the PRF used for the above MSPMK derivation, it can be 128 bits, 160 bits, or 256 bits.

The new key hierarchy is illustrated in Figure 47.

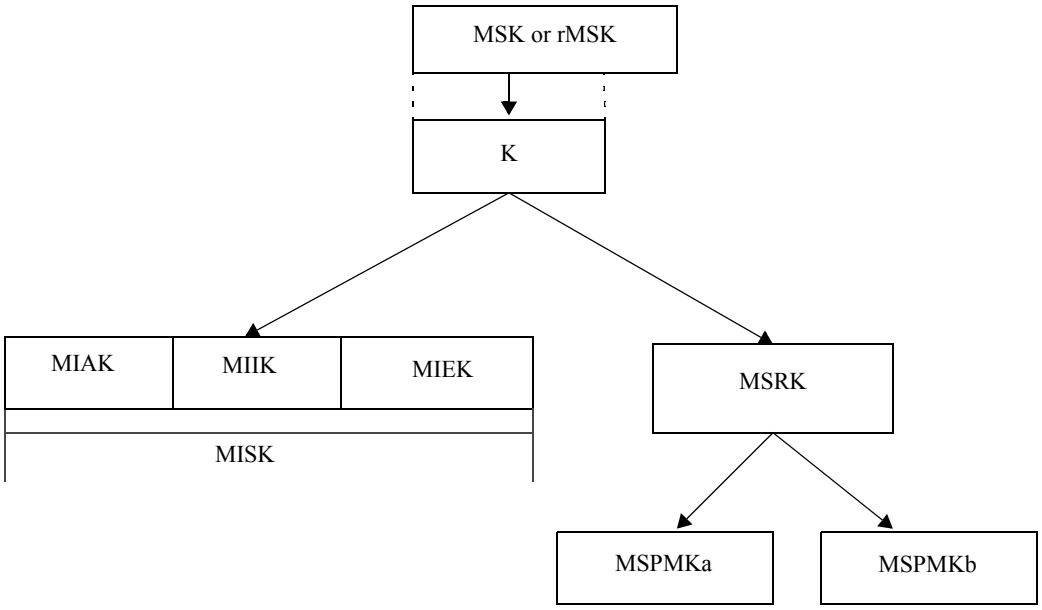



**Figure 47—Key Hierarchy for Bundle Case**

### 10.2.2 Media specific key distribution

Each MSPMK will be distributed to a PoA. The key distribution from the PoS to a PoA can be done through push or pull key distribution. In general, key distribution from a PoS to a PoA is out of the scope of this standard. However, MIH service can be used to trigger the key distribution. The key distribution can be triggered in the following methods.

#### 10.2.2.1 Push key distribution

The objective of push key distribution is to trigger a PoS to push a key into a target PoA. To perform the installation, the MN uses the MIH protocol, which at this point could be protected, to notify the PoS to start the key installation. In the PoS, the key is pushed from MIHF to a MIH user for the further distribution to a PoA. The primitives for push key distribution are defined in 7.4.27. The messages are defined in 8.6.1.16 and 8.6.1.17.

#### 10.2.2.2 Reactive pull key distribution

A reactive pull key distribution is performed after the MN moves to the target PoA. Since no MIH function is used, this is out of the scope of this standard.

#### 10.2.2.3 Optimized proactive pull key distribution

This mechanism allows the MN to perform a media-specific authentication proactively with a target PoA without being directly connected to the wireless link of the target PoA by means of sending link-layer frames through the PoS to the target PoA. The key hierarchy shared between the MN and the PoS is used in order to derive a pre-shared key to conduct a proactive authentication. The PoS is acting as a local authentication server (AS). The PoA receiving the link-layer frames with the authentication information can contact with the AS (the PoS) using the identifier provided during the service access authentication. Once

the proactive authentication is completed, a media specific master session key (MSK) is distributed from the PoS (acting as an AS) to the PoA. At the end, the MN and the PoA share the same media specific MSK. To perform this key distribution mechanism the primitives are defined in 7.4.28 and MIH messages are defined in 8.6.1.18 and 8.6.1.19.

# Annex A

(informative)

# Bibliography

***Insert bibliographical reference as follows:***

IETF RFC 5677 (2009) IEEE 802.21 Mobility Services Framework Design (MSFD).

# Annex D

(normative)

# Mapping MIH messages to reference points

***Insert the following rows to Table D.1*:**

**Table D.1—Mapping MIH messages to reference points**

| MIH message name | Reference point |
|---|---|
| MIH_Auth | RP1, RP3 |
| MIH_Termination_Auth | RP1, RP3 |
| MIH_Push_Key | RP1, RP3 |

# Annex F

(normative)

# Data type definition

## F.3 Derived data types

### F.3.2 General data type

***Change the following row in Table F.2:***

**Table F.2—General data types**

| Data type name | Derived from | Definition |
|---|---|---|
| STATUS | ENUMERATED | The status of a primitive execution.<br><br>0: Success<br>1: Unspecified failure<br>2. Rejected<br>3. Authorization failure<br>4. Network error<br>5. Authentication Failure |

### F.3.12 Data type for MIH capabilities

***Change the following row in Table F.20:***

**Table F.20—Data type for MIH capabilities**

| Data type name | Derived from | Definition |
|---|---|---|
| MIH_IQ_TYPE_LST | BITMAP(64) | Bit 29:<br>IE_AUTHENTICATOR_LINK_ADDR<br>Bit 30: IE_AUTHENTICATOR_IP_ADDR<br>Bit 31: IE_PoS_IP_ADDR<br>Bit 32: IE_KEY_DIST_INF<br>Bit 33: IE_PoS_INTG_ALG_INF<br>Bit 34: IE_PoS_ENCR_ALG_INF<br>Bit 35: IE_PoS_PRF_INF<br><br>Bit ~~29~~ 36– 63 (Reserved) |

***Insert F.3.16 after F.3.15 as follows:***

## F.3.16 Data type for security

### F.24 Data type for security

| Data type | Derived from | Definition |
|---|---|---|
| ID_TYPE | EUMERATED | The type of security association.<br><br>0: TLS-generated;<br>1: EAP-generated |
| ID_VALUE | OCTET_STRING | Represents a security association identifier |
| MIH_SEC_CAP | SEQUENCE(<br>TLS_CAP,<br>EAP_CAP) | Represents the MIH security capabilities. |
| TLS_CAP | BOOLEAN | TLS-generated SA capability.<br>TRUE: (D)TLS Supported<br>FALSE: (D)TLS not supported |
| EAP_CAP | CHOICE ( NULL, SEQUENCE<br>(KEY_DIST_LIST,<br>INT_ALG_LIST,<br>CIPH_ALG_LIST,<br>PRF_LIST)) | EAP-generated SA capability. When NULL is chosen, EAP-generated SA is not supported. When SEQUENCE is chosen, EAP-generated SA is supported. |
| KEY_DIST_LIST | BITMAP(B) | Represents a list of key distribution methods.<br>Bitmap values<br>Bit 0: Push key distribution<br>Bit 1: Optimized proactive pull key distribution<br>Bit 2: Reactive pull key distribution<br>Bit 3–7 (Reserved) |
| INT_ALG_LIST | BITMAP(B) | Represents a list of integrity algorithm.<br>Bitmap values<br>Bit 0: INTG_HMAC_SHA_96<br>Bit 1: INTG_HMAC_CMAC_AES<br>Bit 2–7 (Reserved) |
| CIPH_ALG_LIST | BITMAP(B) | Represents a list of encryption algorithm.<br>Bitmap values<br>Bit 0: ENCR_AES_CBC<br>Bit 1: AUTH_ENC_AES_CCM<br>Bit 2: ENCR_NULL<br>Bit 3-7 (reserved) |
| PRF_LIST | BITMAP(B) | Represents a list of key derivation functions.<br>Bitmap values<br>Bit 0: PRF_AES_CMAC<br>Bit 1: PRF_HMAC_SHA1<br>Bit 2: PRF_HMAC_SHA256<br>Bit 3-7 (reserved) |
| NONCE_VALUE | UNSIGNED_INT(2) | Represents a random value |

## F.24 Data type for security *(continued)*

| Data type | Derived from | Definition |
|---|---|---|
| AUTH_INFO_VALUE | OCTET_STRING | Represents the authentication information used to authenticate. |
| AUTH_VALUE | OCTET_STRING | Represents a message authentication/integrity code. |
| KEY | OCTET_STRING | Represents a cryptographic key. |
| KEY_MAPPING | LIST(SEQUENCE(LINK_TUPLE_ID, KEY, LIFETIME)) | Represents a map of a link layer identifier of a PoA to a key and a lifetime. |
| LINK_AUTHENTICATOR_LIST | SEQUENCE(LINK_TYPE, LIST(LINK_ADDR)) | Represents a list of link layer addresses of authenticators for a given link type. |
| LL_FRAMES | OCTET_STRING | Represents the information needed to carry out a key installation. |
| LIFETIME | UNSIGNED_INT(2) | Represents the period of time that a key is valid and can be used. |
| SECURITY | CHOICE(TLS_RECORD, MIH_SPS_RECORD) | Represents information which is carried in the security TLV. |
| TLS_RECORD | OCTET_STRING | Represents a TLS record. |
| MIH_SPS_RECORD | SEQUENCE(ENCR_BLOCK, CHOICE(INTG_BLOCK, NULL)) | Represents data protected by an MIH security association. |
| ENCR_BLOCK | OCTET_STRING | Represents encrypted data. |
| INTG_BLOCK | OCTET_STRING | Represents integrity protected data. |

# Annex G

(normative)

# Information element identifiers

***Insert the following rows in Table G.1***:

**Table G.1—Information element identifier values**

| Name of information element or container | IE Identifier |
|---|---|
| IE_AUTHENTICATOR_LINK_ADDR | 0x10000206 |
| IE_AUTHENTICATOR_IP_ADDR | 0x10000207 |
| IE_PoS_IP_ADDR | 0x10000208 |

# Annex H

(normative)

# MIIS basic schema

***Change the following text shown:***

The following text defines the RDF vocabularies for MIIS.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY mihbasic "URL_TO_BE_ASSIGNED#">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>

<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;"
xmlns:mihbasic="&mihbasic;" xml:base="&mihbasic;"
xmlns:owl="&owl;" xmlns:xsd="&xsd;">

<owl:Class rdf:ID="POA">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_poa_link_addr"/>
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_poa_location"/>
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_poa_channel_range"/>
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_poa_system_info"/>
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_poa_subnet_info"/>
<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:minCardinality>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_authenticator_link_addr"/>
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_authenticator_ip_addr"/>
<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#ie_pos_ip_addr"/>
<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:comment>
This class contains all the information depicting a PoA.
</rdfs:comment>
</owl:Class>
```

***Insert the following prior to* </rdf:RDF>:**

```
<owl:ObjectProperty rdf:ID="ie_authenticator_link_addr">
<mihbasic:ie_type_identifier>0x10000206</mihbasic:ie_type_identifier>
<rdfs:domain rdf:resource="#POA"/>
<rdfs:range rdf:resource="#TRANSPORT_ADDR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ie_authenticator_ip_addr">
<mihbasic:ie_type_identifier>0x10000207</mihbasic:ie_type_identifier>
<rdfs:domain rdf:resource="#POA"/>
<rdfs:range rdf:resource="#TRANSPORT_ADDR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ie_pos_ip_addr">
<mihbasic:ie_type_identifier>0x10000208</mihbasic:ie_type_identifier>
<rdfs:domain rdf:resource="#POA"/>
<rdfs:range rdf:resource="#TRANSPORT_ADDR"/>
</owl:ObjectProperty>
```

# Annex J

(informative)

# IEEE 802.21 MIB

## J.2 IEEE 802.21 MIB definition

***Change the following text as shown*:**

```
-- ************************************************************************
-- * MODULE IDENTITY
-- ************************************************************************

ieee802dot21 MODULE-IDENTITY
LAST-UPDATED "~~200806041455Z~~201105161205Z"
ORGANIZATION "IEEE 802.21"
CONTACT-INFO
"WG E-mail: stds-802-21@ieee.org
Chair: ~~Vivek G. Gupta~~ Subir Das
~~Intel Corporation~~ Telcordia Technologies
E-mail: ~~mailto:vivek.g.gupta@intel.com~~ subir@research.telcordia.com
Editor: ~~Qiaobing Xie~~ David Cypher
E-mail: ~~Qiaobing.Xie@MOTOROLA.COM~~ david.cypher@nist.gov"
DESCRIPTION
"The MIB module for IEEE 802.21 entities.
iso(1).std(0).iso8802(8802).ieee802dot21(21)"
REVISION "~~200806041455Z~~201105161205Z"
DESCRIPTION
"The latest version of this MIB module."
::= { iso std(0) iso8802(8802) ieee802dot21(21) }
```

***Change the following text as shown:***

```
Dot21ISQueryTypeList ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
" This attribute will be a set of supported MIH IS query types."
REFERENCE "IEEE Std 802.21, 2008 Edition, F.3.12"
SYNTAX BITS
{ binary(0),
rdfData(1),
rdfSchemaUrl(2),
rdfSchema(3),
typeIeNetworkType(4),
typeIeOperatorIdentifier(5),
typeIeServiceProviderIdentifier(6),
typeIeCountryCode(7),
typeIeNetworkIdentifier(8),
typeIeNetworkAuxiliaryIdentifier(9),
typeIeRoamingPartners(10),
typeIeCost(11),
typeIeNetworkQos(12),
typeIeNetworkDataRate(13),
typeIeNetworkRegulatoryDomain(14),
```

```
typeIeNetworkFrequencyBands(15),
typeIeNetworkIpConfigurationMethods(16),
typeIeNetworkCapabilities(17),
typeIeNetworkSupportedLcp(18),
typeIeNetworkMobilityManagementProtocol(19),
typeIeNetworkEmergencyServiceProxy(20),
typeIeNetworkImsProxyCscf(21),
typeIeNetworkMobileNetwork(22),
typeIePoaLinkAddress(23),
typeIePoaLocation(24),
typeIePoaChannelRange(25),
typeIePoaSystemInformation(26),
typeIePoaSubnetInformation(27),
typeIePoaIpAddress(28), }
typeIeAuthenticatorLinkAddress(29),
typeIeAutheticatorIpAddress(30),
typeIePosIpAddress(31) }
```

# Annex K

(informative)

# Example MIH message fragmentation

***Insert the subclause heading before existing text as shown:***

## K.1 Example of original MIH message frangmentation

***Insert K.2 as follows*:**

## K.2 Calculation of securityOverhead when there is an MIH SA

To calculate securityOverhead when there is an MIH SA, the following parameters are used:

- x is 0 when Source MIHF Identifier TLV and Destination MIHF Identifier TLV are contained in the protected MIH message, otherwise, x is 1.
- y is 1 for TLS-generated MIH SA. Otherwise, y is 0.
- $L_{SAID}$ denotes the octet length of the SAID TLV carried in the protected MIH message. $L_{SAID}$ depends on the implementation.
- $L_{SID}$ denotes the octet length of the Source MIHF Identifier TLV optionally carried in the protected MIH message. $L_{SID}$ depends on the implementation.
- $L_{DID}$ denotes the octet length of the Destination MIHF Identifier TLV optionally carried in the protected MIH message. $L_{DID}$ depends on the implementation.
- $O_{SECTLV}$ denotes the overhead of the Security TLV carried in the protected MIH message.
- $O_{TYPE}(y)$ denotes the overhead of the MIH data type contained in the Security TLV.
- $O_{TLS}$ denotes the overhead of the TLS record. $O_{TLS} = 5$, i.e., 1-octet TLSCiphertext.type plus 2-octet TLSCiphertext.version plus 2-octet TLSCiphertext.length [RFC5246].
- $O_{ENC}$ denotes the overhead of encryption. $O_{ENC}$ depends on the ciphersuite.
- $O_{INTG}$ denotes the overhead of integrity protection. $O_{INTG}$ depends on the ciphersuite.

securityOverhead is calculated as follows:

$$\text{securityOverhead} = L_{SAID} - x*(L_{SID} + L_{DID}) + O_{SECTLV} + O_{TYPE}(y) + y*O_{TLS} + O_{ENC} + O_{INTG}$$

Note that securityOverhead can be a negative value when x = 1.

Since the maximum size of Security TLV is no more than the maximum size of Variable Payload of MIH message, which is $2^{16}-1$ octets, the maximum values of OSECTLV and OTYPE(y) are shown below.

- $O_{SECTLV} = 3$ (i.e., 1-octet TLV Type plus 2-octet TLV Length).
- $O_{TYPE}(0) = 6$, i.e., 1-octet CHOICE Selector in CHOICE(TLS_RECORD, MIH_SPS_RECORD) plus 2-octet Length field of ENCR_BLOCK data plus 1-octet CHOICE Selector in MIH_SPS_RECORD plus 2- octet Length field of INTG_BLOCK data.
- $O_{TYPE}(1) = 3$, i.e., 1-octet CHOICE Selector in CHOICE(TLS_RECORD, MIH_SPS_RECORD) plus 2-octet Length field of TLS_RECORD data.

Table K.1 shows $O_{ENC}$ and $O_{INTG}$ values for the MIH ciphersuites for EAP-generated MIH SA.

**Table K.1—Protection Overhead for EAP-generated SAs**

| Ciphersuite code | Encryption | Integrity Ptotection | $O_{ENC}$ | $O_{INTG}$ |
|---|---|---|---|---|
| 00000010 | AES_CBC | HMAC-SHA1-96 | 32(IV+padding) | 12 (MIC) |
| 00000100 | NULL | HMAC-SHA1-96 | 0 | 12 (MIC) |
| 00000101 | NULL | AES_CMAC | 0 | 12 (MIC) |
| 00000110 | AES_CCM | | 10 (SN)+ 12(MIC) | 0 |

For example, consider a case where Ciphersuite Code 00000010 (AES-CBC + HMAC-SHA1-96) is used for EAP-generated MIH SA (y=0) without containing Source MIHF Identifier TLV and Destination MIHF Identifier TLV in the protected MIH message (x=0), and the length of SAID TLV, the length of Source MIHF Identifier TLV, the length of Destination MIHF Identifier TLV are 30 octets, 20 octets and 30 octets, respectively. Then securityOverhead is computed as:

securityOverhead = $L_{SAID}$ – ($L_{SID}$ + $L_{DID}$) + $O_{SECTLV}$ + $O_{TYPE}$(0) + $O_{ENC}$ + $O_{INTG}$

= 30– (20+30)+3+6+44 = 33 (octets).

Figure K.2 shows the protected fragments for the original message shown in Figure K.1, when operating in the same condition as described in the above example with securityOverhead=33 (octets). The integer number within the brackets of each field in Figure K.2 indicates the length of the field in octets. In Figure K.2, the fragment size before applying MIH protection is set to 1424 (=16*89) octets to have the fragment size of 1499 octets after applying MIH protection, which gives the largest number of 16-octet blocks (89) under the condition that the resulting protected fragment does not exeeds 1500 octets.

First protected fragment message (M=1, FN=0, size =1499 octets)

| Header (S=1) (8) | SAID TLV (30) | Security TLV (1461) |
|---|---|---|

Encrypted fragment = 16*19 = 1424 octets
IV = 16 octets
MIC = 12 octets
TLV overhead = 3 octets
MIH data type overhead = 6 octets

Second protected fragment message (M=0, FN=1, size = 251 octets)

| Header (S=1) (8) | SAID TLV (30) | Security TLV (213) |
|---|---|---|

Encrypted fragment = 1600-1424 = 176 octets
IV = 16 octets
MIC = 12 octets
TLV overhead = 3 octets
MIH data type overhead = 6 octets

**Figure K.2—Example of protected MIH fragment message**

# Annex L

(normative)

## MIH protocol message code assignment

***Insert the following rows into Table L.1 as shown:***

**Table L.1—AID assignment**

| Messages | AID |
|---|---|
| MIH messages for service management | |
| MIH_Auth | 6 |
| MIH_Termination_Auth | 7 |
| MIH_Push_Key | 8 |
| MIH_LL_Auth | 9 |

***Insert and change the following rows in Table L.2 as shown:***

**Table L.2—Type values for TLV encoding**

| TLV type name | TLV type value | Data type |
|---|---|---|
| (Reserved) | ~~64~~80-99 | (Reserved) |
| Security | 64 | SECURITY |
| SAID | 65 | SEQUENCE(ID_TYPE, ID_VALUE) |
| Security capability | 66 | MIH_SEC_CAP |
| KeyLifeTime | 67 | LIFETIME |
| AUTH | 68 | AUTH_VALUE |
| NONCE | 69 | NONCE_VALUE |
| Authentication | 70 | AUTH_INFO_VALUE |
| Link identifier | 71 | LINK_TUPLE _ID |
| Link layer information | 72 | LL_FRAME |
| Link Tuple Identifier List | 73 | LIST(LINK_TUPLE_ID) |
| Authenticator list | 74 | LIST(LINK_AUTHENTICATOR_LIST) |
| Ciphersuite | 75 | SEQUENCE(KEY_DIST_LIST, INT_ALG_LIST, CIPH_ALG_LIST, PRF_LIST) |

# Annex M

(normative)

# Protocol implementation conformance statement (PICS) proforma[6]

## M.8 PICS proforma tables

### M.8.3 Major capabilities

***Add the following rows into the table in M.8.3 as shown:***

| Item number | Item description | Reference | Status | Support | Mnemonic |
|---|---|---|---|---|---|
| M.8.3.9a | Is MIH service access authentication supported? | 6.2.1, 9 | M | Yes [ ] No [ ] | MC9a |
| M.8.3.9b | Is MIH message protection supported? | 9 | M | Yes [ ] No [ ] | MC9b |
| M.8.3.9b.1 | Is EAP supported? | 9.2, 9.3 | O.x<br>MC9a or mc9B<br>M:<br>MC9c.2 | Yes [ ] No [ ] | MC9b.1 |
| M.8.3.9b.1.1 | Is AES_CCM (ciphersuite 00000110) supported? | 9.2.3, 9.3.2 | M:<br>MC9b.1 | Yes [ ] No [ ] | |
| M.8.3.9b.1.2 | Is AES_CBC and HMAC-SHA1-96 (ciphersuite 00000010) supported? | 9.2.3, 9.3.3 | O:MC9b.1 | Yes [ ] No [ ] | |
| M.8.3.9b.1.3 | Is HMAC-SHA1-96 (cyphersuite 00000100) supported? | 9.2.3, 9.3.4 | O:MC9b.1 | Yes [ ] No [ ] | |
| M.8.3.9b.1.4 | Is AES-CMAC (cyphersuite 00000101) supported? | 9.2.3, 9.3.5 | O:MC9b.1 | Yes [ ] No [ ] | |

[6]Copyright release for PICS proformas: Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

| Item number | Item description | Reference | Status | Support | Mnemonic |
|---|---|---|---|---|---|
| M.8.3.9b.2 | Is (D)TLS supported? | 9.1 | O.x<br>MC9a or MC9b | Yes [ ] No [ ] | MC9b.2 |
| M.8.3.9c | Is proactive authentication supported? | 5.1.9, 10 | O | Yes [ ] No [ ] | MC9c |
| M.8.3.9c.1 | Is unbundled proactive authentication supported? | 10.1 | O.w:MC9c | Yes [ ] No [ ] | MC9c.1 |
| M.8.3.9c.2 | Is bundled proactive authentication supported? | 10.2 | O.w:MC9c | Yes [ ] No [ ] | MC9c.2 |

***Insert new Annex N and Annex O as follows:***

# Annex N

(informative)

# Authentication and key distribution procedures

## N.1 MIH service access authentication

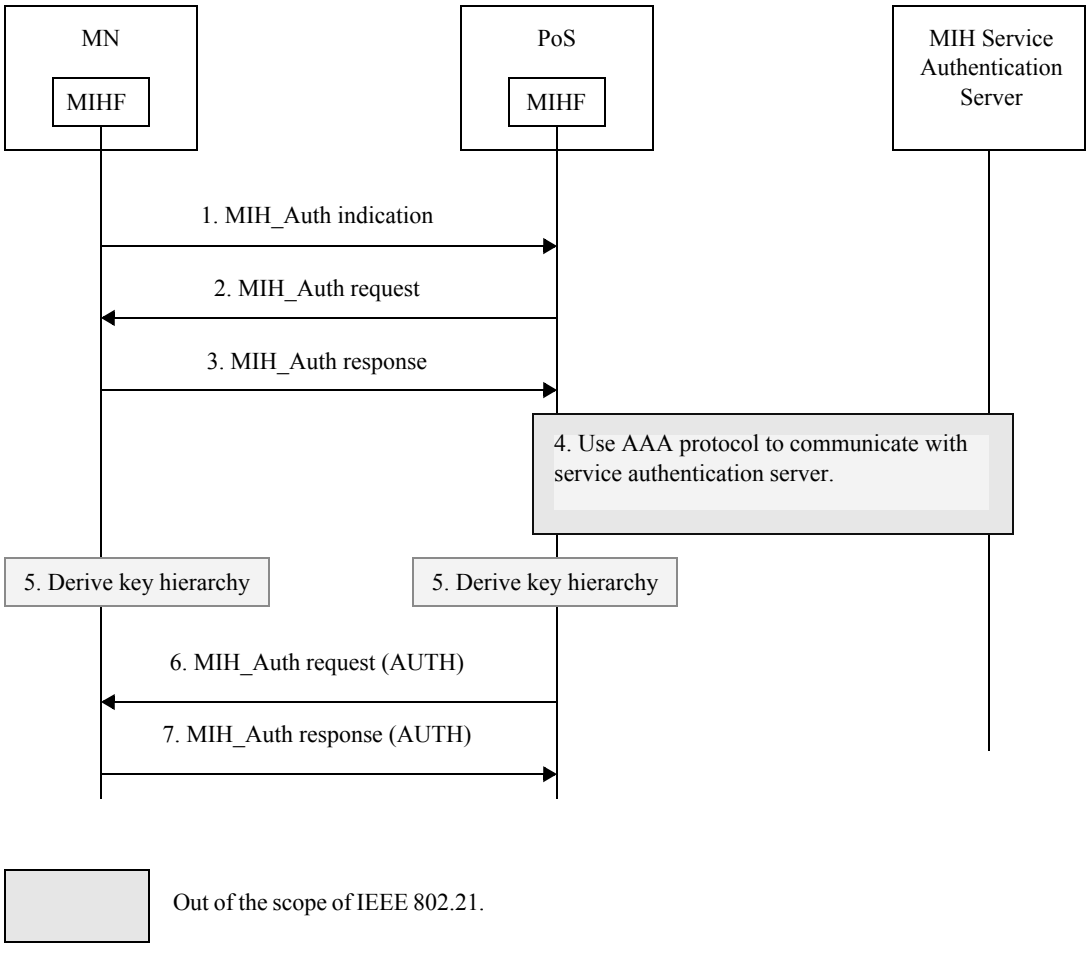



**Figure N.1—Mobile initiated access authentication phase**

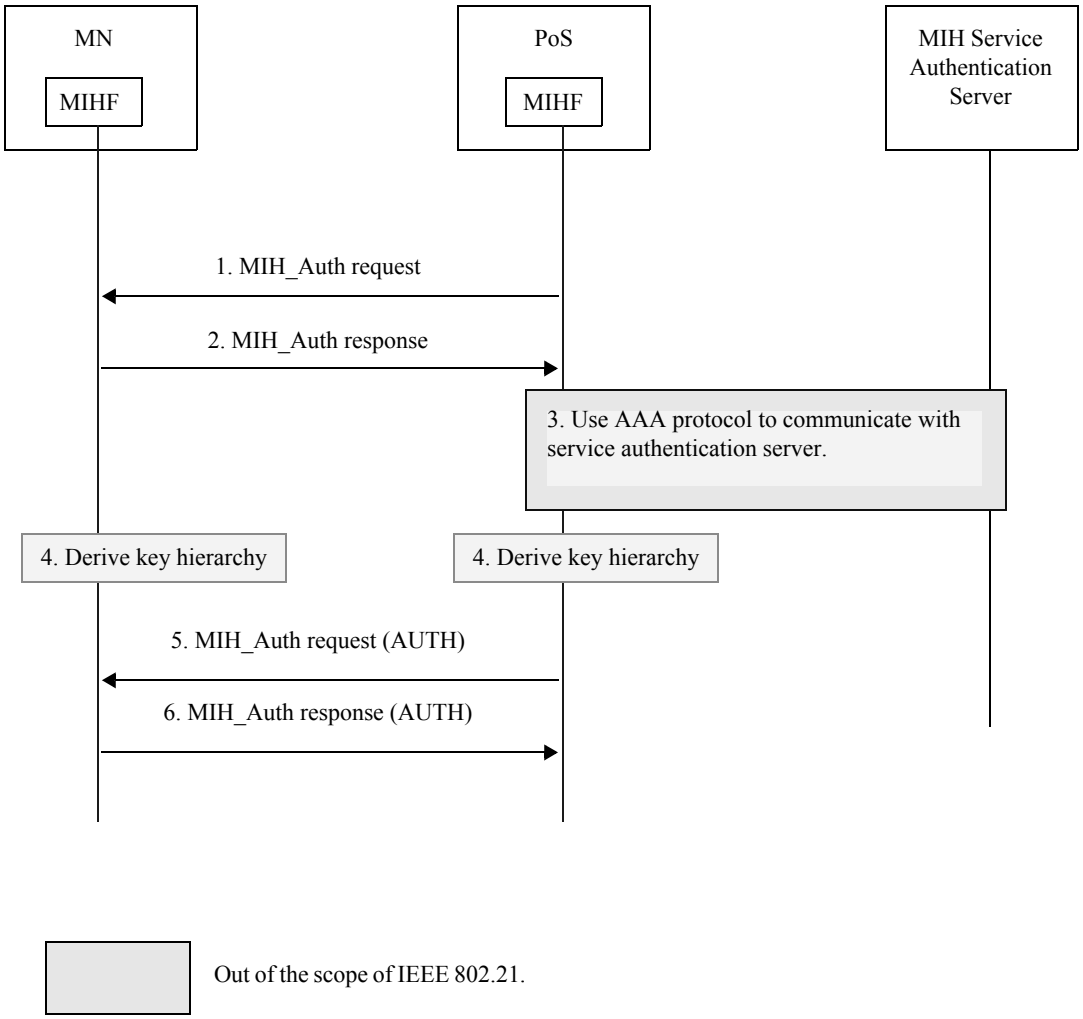



**Figure N.2—Network initiated access authentication phase**

## N.2 Push key distribution

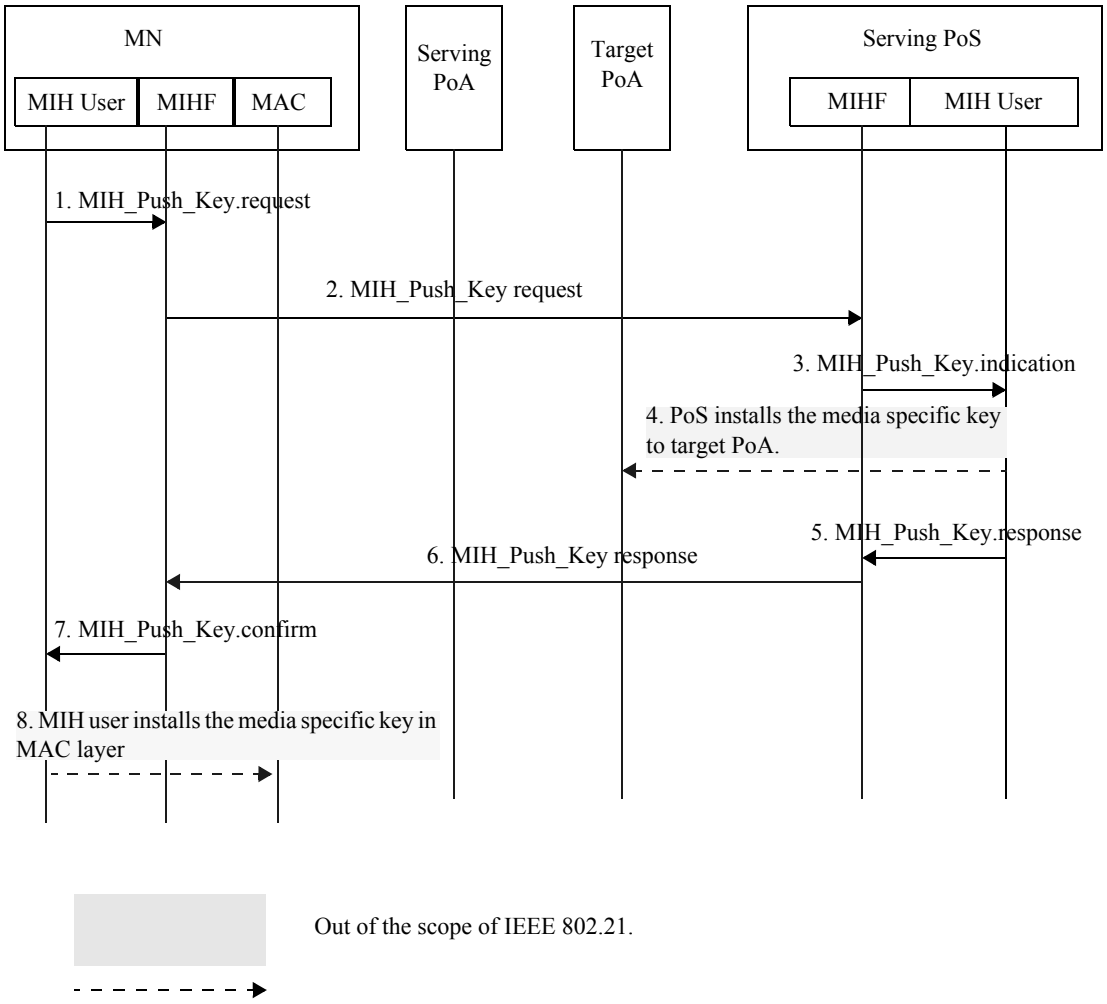



**Figure N.3—Push key distribution**

## N.3 Proactive authentication

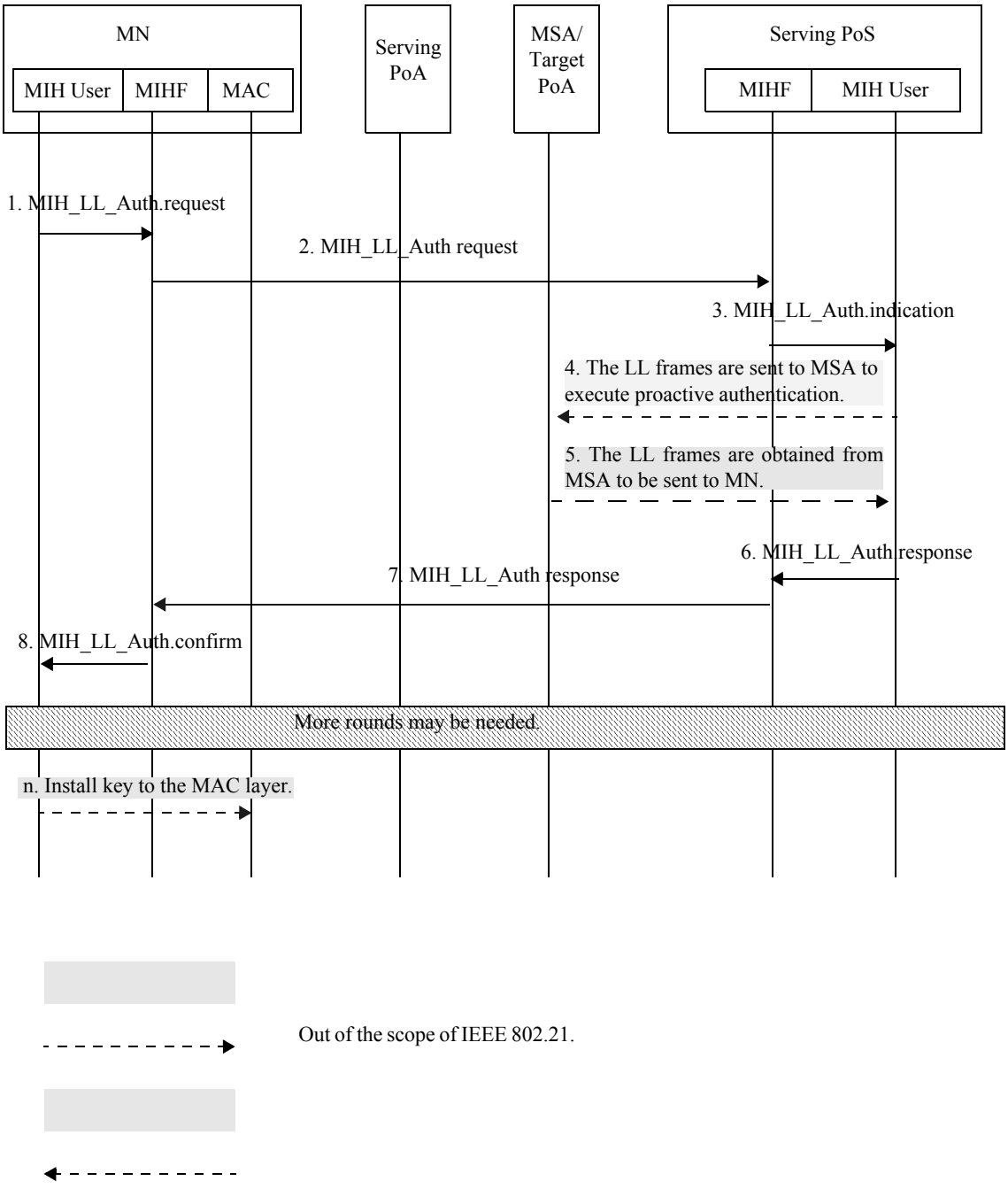



**Figure N.4—Proactive authentication**

## N.4 Optimized pull key distribution

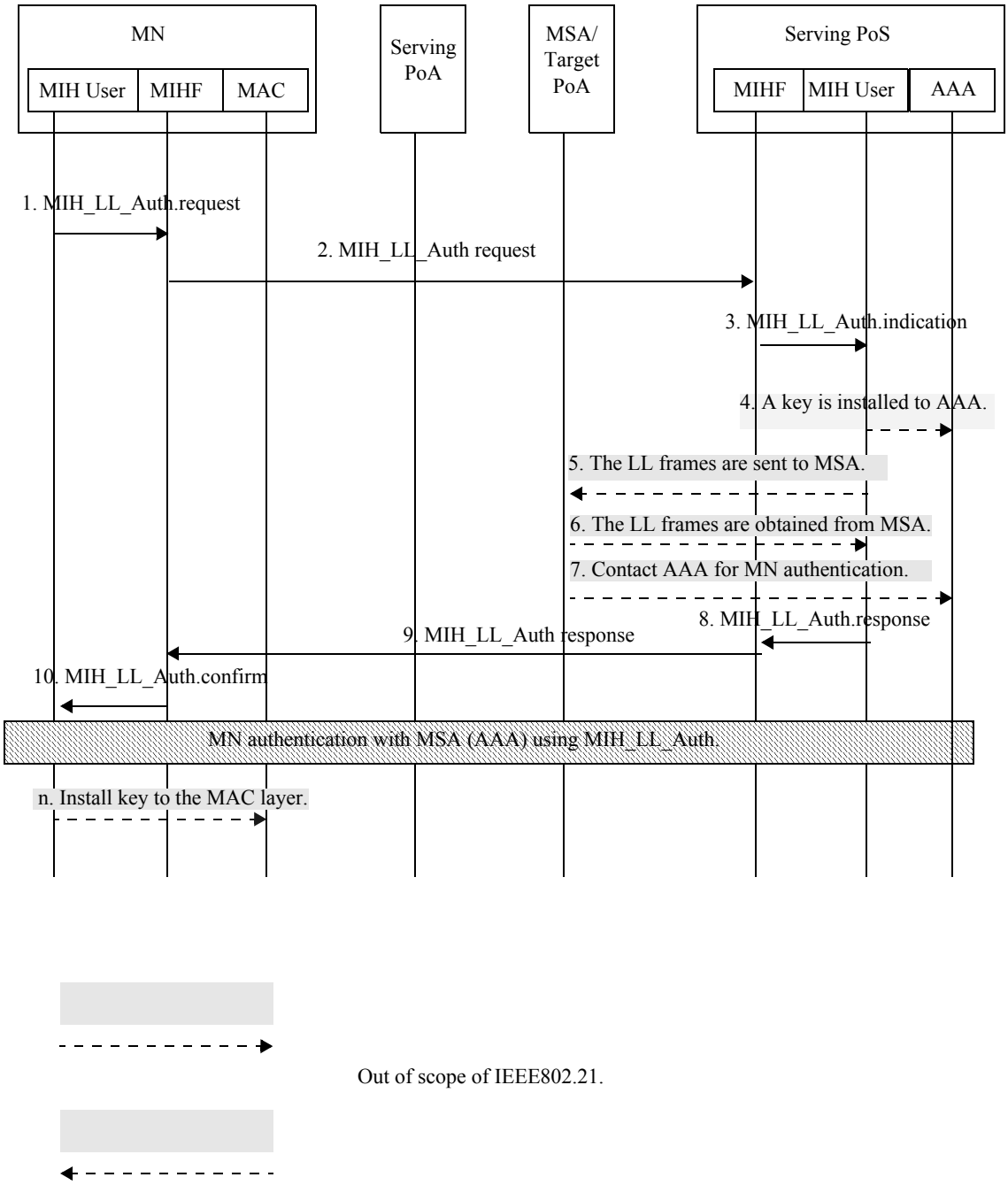



**Figure N.5—Optimized pull key distribution**

## N.6 Termination phase

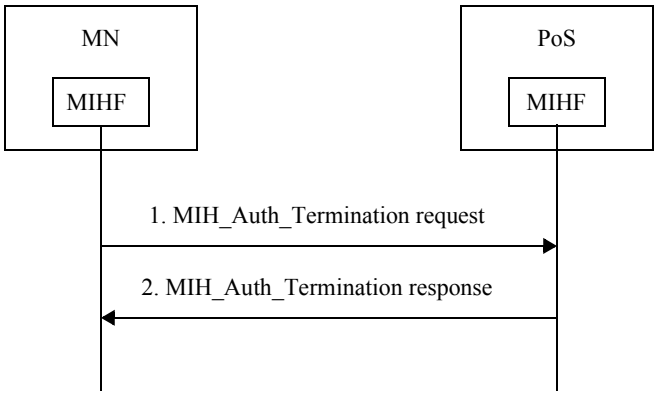



**Figure N.6—MN initiated termination phase**

# Annex O

(informative)

# Protection through transport protocols

MIH messages can be carried over wireless protocols in layer 2 such as defined in IEEE Std 802.11 or layer 3 as defined in IETF RFC 5677. In the following, the security protection provided through the transport protocol are discussed and security issues are identified with each protection mechanism.

## O.1 Protection through layer 2

When MIH messages are transported over a layer 2 protocol, the protection may be provided through the layer 2 protocol such as TKIP and CCMP specified in IEEE Std 802.11.

The protection in layer 2 is usually established with L2 identifiers such as MAC address for an MN and a PoS. MIH messages are protected together with other data. Furthermore, if MIH messages are transported over different layer 2 protocols, then the protection may be different. If the PoS is not co-located with a PoA in the same device, the protection through a L2 protocol may not provide end-to-end security between the MN and the PoS.

On the other hand, such protection through a layer 2 protocol will not require any change on either MIH protocol or the layer 2 protocol that transports the MIH protocol.

## O.2 Protection through IPsec

When MIH messages are transported over IP as defined in IETF RFC 5677, they may be protected by IPsec as specified in IETF RFC 4302 for IP Authentication Header (AH) and RFC 4303 IP Encapsulating Security Payload (ESP). When IPv6 is implemented in a mobile node and a PoS, then IPsec is mandatory. In this case, each MIH message is protected at IP layer as an IP payload in each IPsec packet.

For a pair of IP nodes with fixed IP addresses, the IPsec Security Associations (SAs) are established through Internet Key Exchange (IKEv1 or IKEv2) specified in IETF RFC 2409 and IETF RFC 4306. However, in case of MIH message protection, the IP address of a mobile node may be dynamic. In this case, a protocol suite defined by IETF RFC 4555 - “IKEv2 Mobility and Multihoming Protocol (MOBIKE)” may be used to establish SAs between an MN and a PoS (a.k.a. MoS as defined in IETF RFC 5677).

It is similar to IKEv2, MOBIKE is a heavy weight protocol. The MOBIKE RFC is explicitly defined for tunnel-mode IPSec connections.

IPsec protocols are well defined and can provide proper protection for its IP payload. When SAs are established between an MN and a PoS, they provide end-to-end protection. Using IPsec will not require any changes to either MIH protocol or IPsec.

Similar to protection provided in layer 2, the protection through IPsec are not MIH specific. However, for the mutual authentication through MOBIKE, the certificates may be issued on identifiers that are related to MIH applications. From this point of view, IPsec is closer to MIH specific protection, compared to L2 protection.