|  |  |  |
| --- | --- | --- |
| Project | **IEEE 802.21 MIHS**  **<**[**http://www.ieee802.org/21/**](http://www.ieee802.org/21/)**>** | |
| Title | **Draft for changing AES-CCM in the GKB-generated MIH SA** | |
| DCN | 21-15-0061-06-REVP | |
| Date Submitted | **11 November, 2015** | |
| Source(s) | Yoshikazu Hanatani (Toshiba) |  |
| Re: | IEEE 802.21m Teleconference | |
| Abstract | Based on the proposal from Panasonic (DCN 21-15-0051-03-REVP) in IEEE 802.21 Session #68, this document provides a draft to reflect the proposal to IEEE 802.21m draft standard. | |
| Purpose | To improve security using the AEC-CCM in the GKB-generated MIH SA in IEEE 802.21m draft standard | |
| Notice | This document has been prepared to assist the IEEE 802.21 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. | |
| Release | The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that IEEE 802.21 may make this contribution public. | |
| Patent Policy | The contributor is familiar with IEEE patent policy, as stated in [Section 6 of the IEEE-SA Standards Board bylaws](http://standards.ieee.org/guides/opman/sect6.html#6.3) <[http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://127.0.0.1:4664/cache?event_id=757737&schema_id=1&s=5X0vID10lu_E6yrIkWkNd4Wz2H8&q=hancock)> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/faq.pdf> | |

Background

In some use case, the GKB-generate SA without digital signature is useful since PoS and MN are not required to generate and verify ECDSA. So, current IEEE 802.21d specification allows omitting the digital signature.

DCN 21-15-0051-03-REVP pointed out that the GKB-generate SA without digital signature is not secure against an insider attack and an outsider attack.

* Both attackers can rewrite Source Identifier TLV, but anyone cannot detect it since MIH header, Source Identifier TLV, Destination Identifier TLV, SAID TLV are not protected by AES-CCM.

The GKB-generated MIH SA without digital signature can be improved by using the associated data of AES-CCM.

* We can protect MIH header, Source Identifier TLV, Destination Identifier TLV, SAID TLV as the associated data of AES-CCM against the outsider who is not a group member.
* Even if using the above method, we cannot protect them against the insider who is a group member. The digital signature is required to prevent the insider attack.
* The associated data is an input of AES-CCM and it is described in NIST SP 800-38C. The associated data is not encrypted but it is authenticated by AES-CCM.
* IEEE 802.21a uses AES-CCM without the associated data.

Outline of proposed draft

1. To use the associated data in AES-CCM, I added new sub clause to explain MIH PDU protection for group addressed message by AES-CCM.

The associated data is explained in 9.6.3.3.1 as follows.   
“The associated data, A, is not encrypted but authenticated. A is formed by concatenating the MIH Header, the Source MIHF Identifier TLV, the Destination MIHF Identifier TLV as presented in MIH PDU and the SAID TLV with padding”

1. The proposed protection procedure is complicated. So, I added new figure to explain MIH PDU protection procedure with the GKB-generated MIH SA.
2. A new multicast ciphersuite is added to obviously describe a protection method without the digital signature.
3. MULTICAST\_CAP is extended to convey a plurality of the multicast ciphersuites.
4. The proposed draft does not affect IEEE 802.21a.
5. Change order of the multicast ciphsersuite.

**=========================================================================**

**Proposed draft**

**=========================================================================**

**9.6 Group addressed message protection**

***Change the text as follows.***

**9.6.2 Multicast message protection**

Depending on the selected group ciphersuite, an MIH PDU may be encrypted, integrity protected, or protected in both aspects. An example procedure is illustrated in Figure X.

In order to send a group addressed message the MIH User of the PoS with group manager generates a request primitive and delivers it to the local MIHF. Upon receiving the request, the MIHF behaves as follows:

~~a) The MIHF generates a Source MIHF ID TLV based on its own MIHF ID.~~

~~b) The MIHF generates a Destination MIHF ID TLV based on the DestinationIdentifier in the received request.~~

~~c) The MIHF generates MIH Service Specific TLVs or a fragment based on the received request primitive.~~

a) The MIHF generates an MIH request or indication message contained an MIH header, a Source MIHF Identifier TLV, a Destination MIHF Identifier TLV, and Service Specific TLVs or a fragment.

b) Consulting with the *Group Membership Information Base*, the MIHF finds the transport address for the group associated with the DestinationIdentifer in the received request.

d) The MIHF ~~generates an MIH request or indication message.~~ runs PDU protection procedure in Figure X.

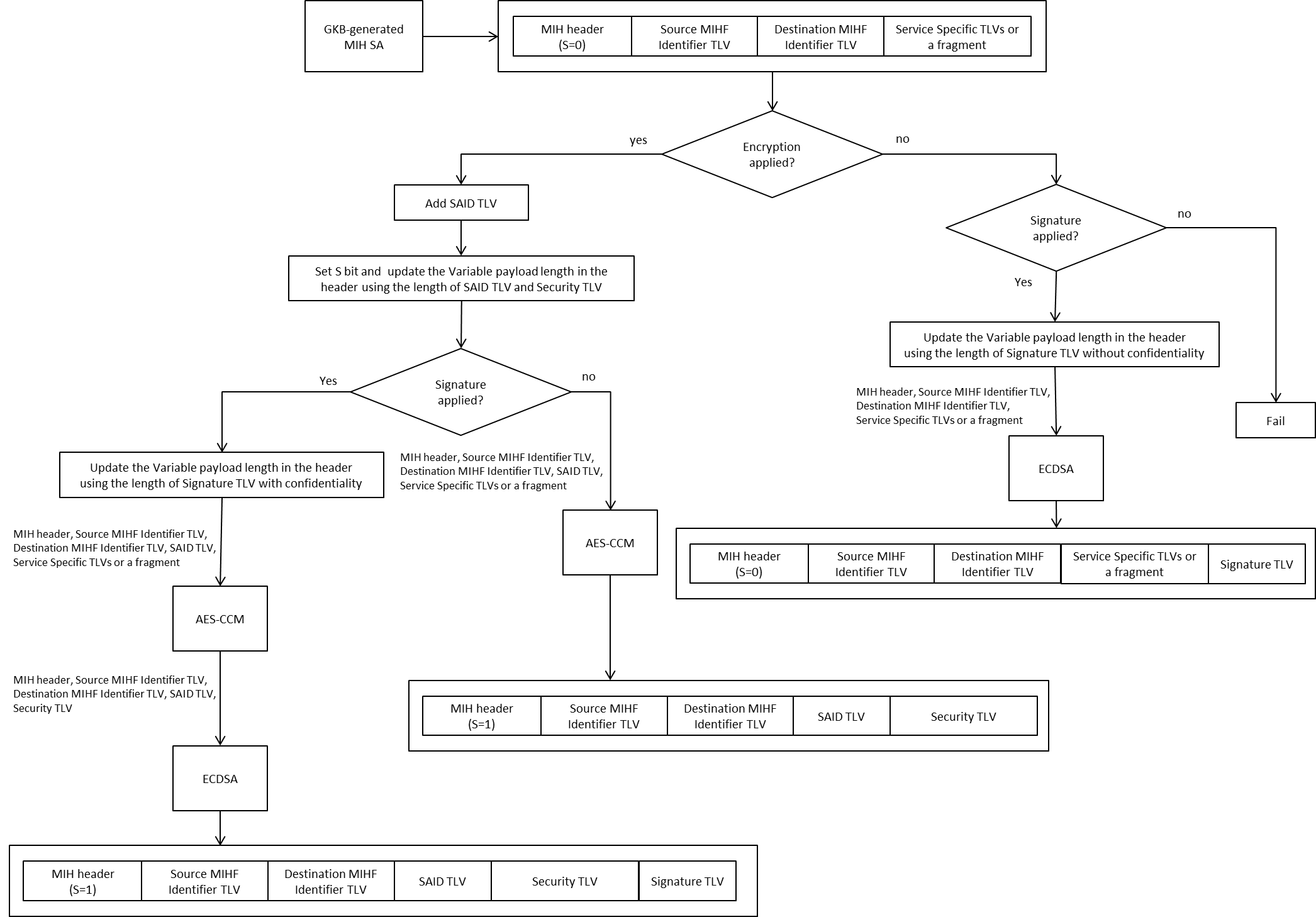
1) The MIH Service Specific TLVs or a fragment may be encrypted with an MIGSK associated to the DestinationIdentifier to make a Security TLV if necessary in the scheme described in 9.6.3~~8.4.2~~.

2) A signature TLV may be generated as shown in 9.6.4.1 using the signing key of the MIHF.

e) The MIHF sends the message to the transport address found in Step d)

The MIH Service Specific TLVs or fragment may be encrypted to make a Security TLV if necessary in the scheme described in 9.6.3~~8.4.2~~.

***Insert new figure as follows:***



**Figure X – MIH PDU protection procedure with the GKB-generated MIH SA**

When an MIHF of a recipient receives the message, the following steps are taken:

a) The Destination Identifier is retrieved from the Destination MIHF ID TLV. The MIHF checks if the Destination Identifier is registered in the Group Information Base or not. If it is not, the message is not for the recipient. Thus, it cancels the following steps and stops processing.

b) The Source Identifier is retrieved from the Source MIHF ID TLV.

c) If the Signature TLV is attached, the MIHF verifies SIGNATURE\_DATA in the Signature TLV using the verification key corresponding with the CERT\_SERIAL\_NUMBER in the Signature TLV as shown in 9.6.4.2.

d) If a Security TLV is contained, the MIHF decrypts the Security TLV in the scheme described in 9.6.3 with the MIGSK associated with the preceding Destination Identifier and the SAID in the Security TLV that is available in the Group Information Base. If the decryption fails, it cancels the following steps and abort. If the decryption succeeds, MIH Service Specific TLVs or a fragment is obtained.

e) If MIH service specific TLVs is obtained or a fragment is obtained and reassembling fragments succeeds, the MIHF issues an indication primitive to its local MIH User.

***Insert following new subclause.***

**9.6.3 MIH PDU protection for group addressed message by AES-CCM**

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 9.6.3.1 through 9.6.3.3.

**9.6.3.1 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).

d) *a* - The length of the binary representation of the octet length of the associated data is 2 octets (16 bits).

**9.6.3.2 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 MIGSK. 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-XX.

****

**Figure 38-XX—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 MIGSK. For a given SA, each of MIHFs keeps an SN, which is the

highest as used for a given MIGSK.

**9.6.3.3 Operational procedures in AES-CCM**

**9.6.3.3.1 Encapsulation**

For a given GKB-generated MIH SA, the prerequisites for AES-CCM encapsulation includes an encryption key MIGSK, an AES block cipher encryption block, and the values of parameters *t*, *n*, *q* and *a*. 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. The associated data, A, is not encrypted but authenticated. A is formed by concatenating the MIH Header, the Source MIHF Identifier TLV, the Destination MIHF Identifier TLV as presented in MIH PDU and the SAID TLV with padding. ~~In this standard, the associated data, A, is null.~~ The data, *P* and *A,* 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-XX

****

**Figure 39-XX—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 one;

— 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-XX



**Figure 40-XX—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 9.6.3.2.

d) Input *N, P* and *A* 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-XX and Figure 40-XX, respectively.

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

**9.6.3.3.2 Decapsulation**

For a given GKB-generated MIH SA, the prerequisites for AES-CCM decapsulation includes an encryption key MIGSK, AES block cipher encryption block, the parameters *t*, *n*, *q* and *a*.

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 9.6.3.2.

d) Input *N, A* 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 9.6.3.3.1.

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

**9.6.2.4 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, A and T (MIC). It is illustrated in Figure 41-XX. Since MIC is carried in the ENCR\_BLOCK data, the INTEGR\_BLOCK in MIH\_SPS\_RECORD is not chosen for AESCCM.



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

***Change the text as follows.***

**9.6.4 Signature and credential management**

In order to enable signing functionality, the message source requests credentials for public key using an out-of-band mechanism that is not specified in this document. The message source provides the credentials to destination devices. Message signing procedure, signature verification procedure and certificate management procedure are described in 9.6.4.1 and 9.6.4.2, respectively.

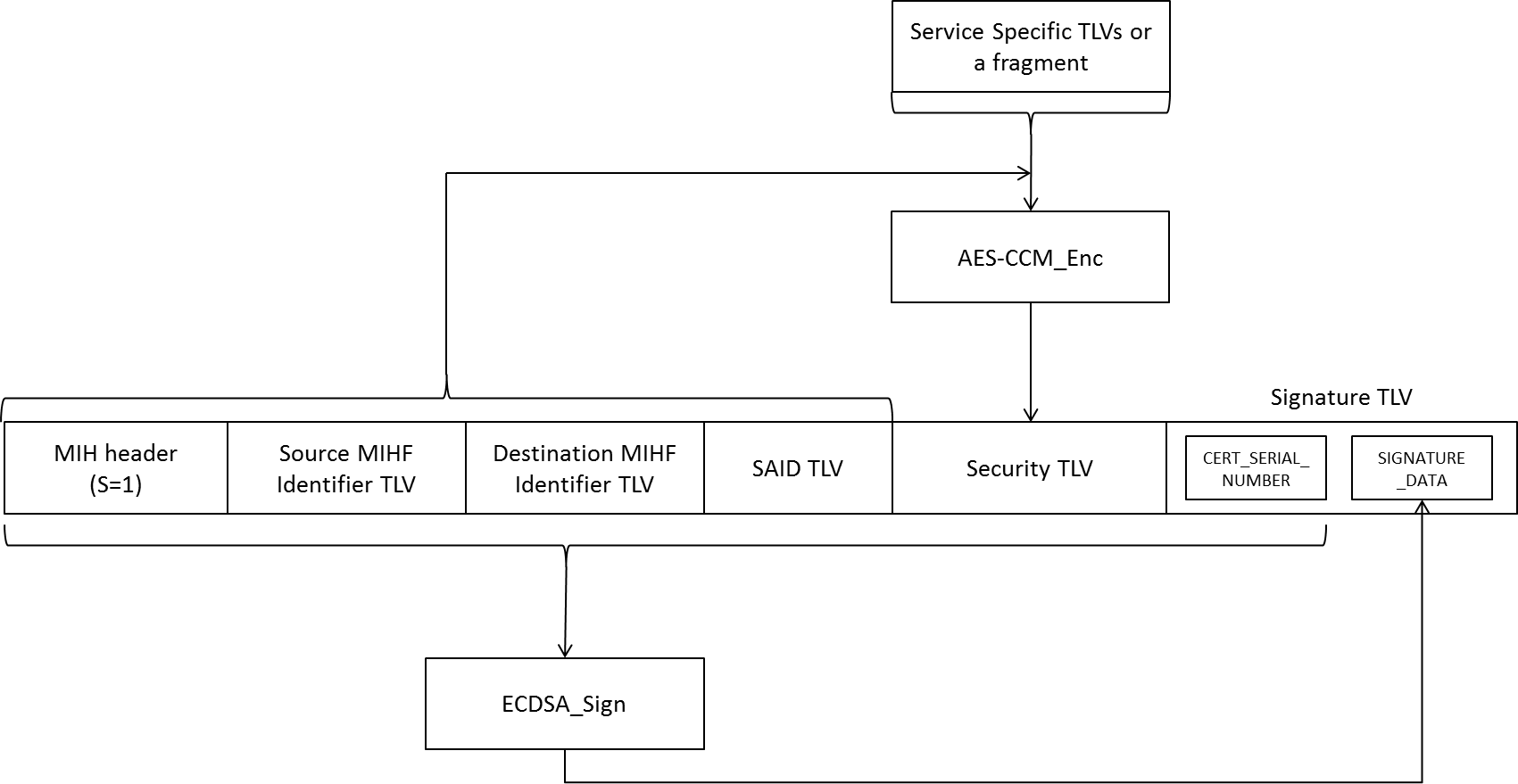
In this specification, Elliptic Curve Digital Signature Algorithm (ECDSA) specified in IEEE 802.1AR- 2009, Secure Device Identity, by reference to NIST FIPS 186-4 and ANSI X9.62-2005 is used as the multicast signature scheme. In particular, NIST recommended elliptic curve P-256 and hash function SHA-256, specified in FIPS 180-4, are used to generate signatures. These algorithm identifiers are defined in Clause 9.6.4.4.

**9.6.4.1 Multicast Message Signatures**

Multicast Messages are signed with the message source using a private key of the message source. Integrity and proof of origin of a multicast message is verified by verifying the message signature with the public key of a message source. The message content is signed using elliptical curve cryptography (the signature algorithm is defined in 9.6.4).

In case the MIH PDU is protected through GKB-generated MIH SA with a signature as specified in 8.4.2.3, the MIHF of PoS generates a Signature TLV consisting of a CERT\_SERIAL\_NUMBER and a SIGNATURE\_DATA. The SIGNATURE\_DATA is created by signing an MIH\_Group\_Manipulate command or a group addressed command using a signing key corresponding with a verification key specified by CERT\_SERIAL\_NUMBER. Figure 49-XX illustrates the data protection procedure with confidentiality.

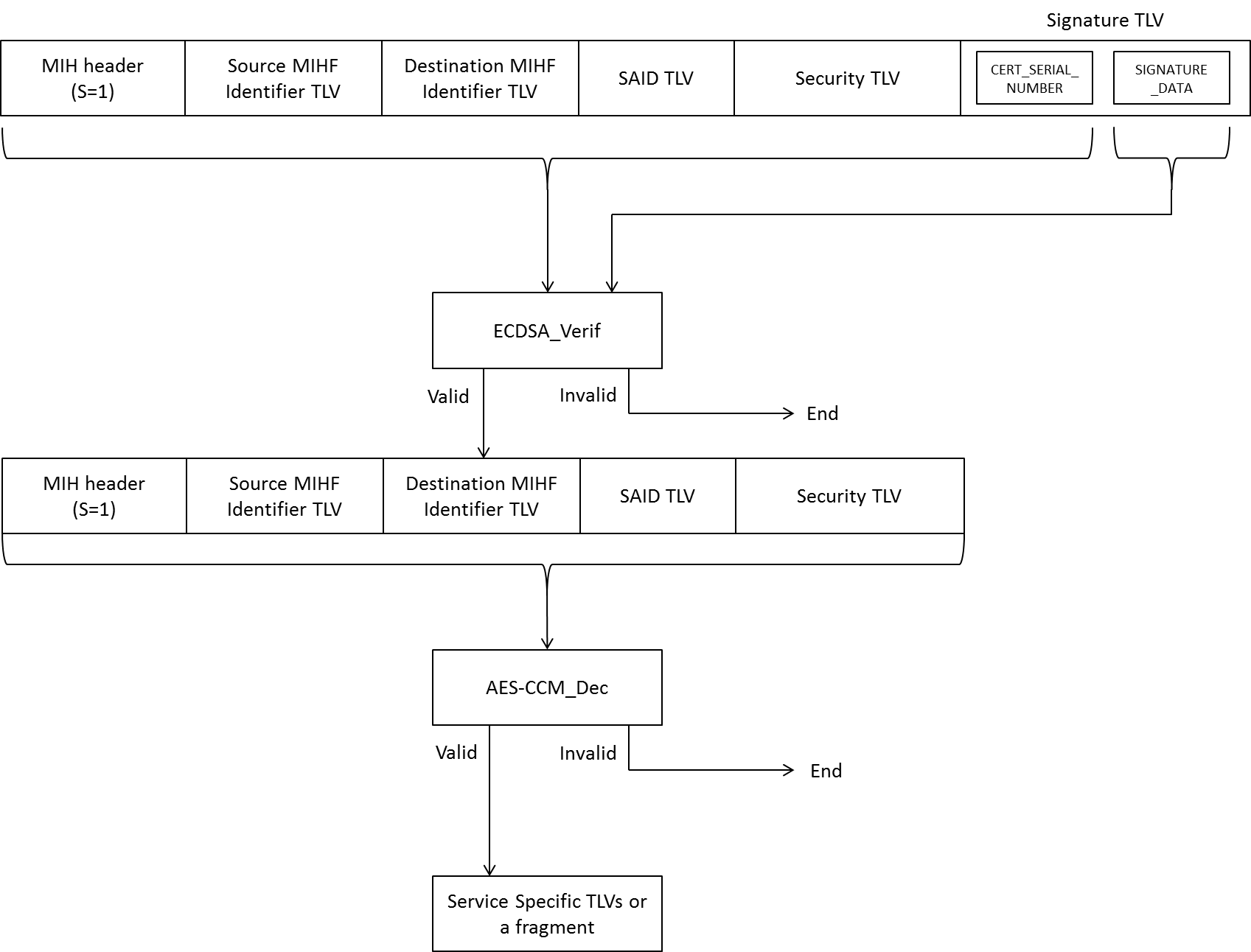
***Change the Figure 49 as follows.***

****

**Figure 49-XX --- Signing (with confidentiality)**

**9.6.4.2 Signature Verification**

***Change the Figure 51 as follows.***

****

**Figure 51-XX --- Signature Verification (with confidentiality)**

***Change the subclause number as follows.***

**9.6.4.3 Certificate Management**

**9.6.4.4 Algorithm identifiers**

**9.6.5 Group ciphersuites**

The ciphersuites used for securing group addressed messages are defined in Table 6.

***Change the Table 6 as follows.***

**Table 6 —Group ciphersuites**

|  |  |  |
| --- | --- | --- |
| **Code** | **Encryption Algorithm** | **Digital Signature Algorithm** |
| 10000100 | NULL | ECDSA-256 |
| 10010001 | AES\_CCM-128 | NULL |
| 10010101 | AES\_CCM-128 | ECDSA-256 |

In Table 6, AES-CCM is an AES mode of operations specified in NIST SP 800-38C. AES-CCM provides confidentiality and data integrity.

In Table 6, ECDSA-256 uses curve P-256 and hash function SHA-256. Notice that AES-CCM uses the group key MIGSK. It can provide data integrity but not unique data origin authentication because the symmetric key is shared among a group of recipients. The data origin authentication is provided through ECDSA.

The support of code “10000100” is mandatory and all entities supporting this specification shall implement it.

***Change the subclause number as follows.***

**9.6.6 Group key distribution Ciphersuites**

**Change following row in Table F.24**

|  |  |  |
| --- | --- | --- |
| MULTICAST\_CAP | SEQUENCE(  LIST(UNSIGNED\_INT(~~2~~1)),  LIST(UNSIGNED\_INT(1)),  IS\_GROUP\_MANAGER) | The first UNSIGNED\_INT represents a  group ciphersuite~~. T~~, the ~~1st~~ octet encodes  a group ciphersuite code defined in ~~0~~9.6.5.  The second UNSIGNED\_INT represents a  group key distribution ciphersuite, the octet encodes a group key distribution cihersuite code defined in ~~0~~9.6.6. |

**Change following row in 7.4.2.3 MIH\_Register.response**

|  |  |  |
| --- | --- | --- |
| MulticastCipherSuite | MULTICAST\_CAP | (Optional) Specifies the group  ciphersuite to be used for securing  multicast MIH messages. ~~Only one~~  ~~ciphersuite shall be included.~~ |

**Change following row in 7.4.2.4 MIH\_Register.confirm**

|  |  |  |
| --- | --- | --- |
| MulticastCipherSuite | MULTICAST\_CAP | (Optional) Specifies the group  ciphersuite to be used for securing  multicast MIH messages. ~~Only one~~  ~~ciphersuite shall be included.~~ |

**Change following text in 8.4.2**

A protected MIH PDU is an MIH PDU that has an MIH header with S bit set to one indicating that the MIH service specific TLVs in this PDU are ~~protected~~ encrypted and/or the PDU is digitally signed. When the MIH service specific TLVs in this PDU are encrypted, each security association is defined for a pair or group of MIHFs 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.~~ In this case, an SAID TLV shall be carried in the PDU. Source and Destination MIHF Identifier TLVs may not be present when an SA is defined for a pair of MIHFs. When a PDU is digitally signed, a Signature TLV shall be carried in the PDU. A Signature TLV should be used for multicast MIH messages in order to provide source origin authentication for multicast MIH messages. Otherwise, a message alternation attack by an insider who has a GKB SA is possible even if the multicast MIH message is integrity protected by the group key corresponding to the GKB SA. On the other hand, a message alternation attack by an outsider who does not have a GKB SA is not possible if the multicast MIH message is integrity protected by the GKB SA but is not protected by a Signature TLV. Especially, a message protected by a GKB SA for a two-member group provides the source origin authentication without a Signature TLV. ~~is not needed for a message that is protected by a GKB SA for a two-member group.~~ Figure 36 shows a protected MIH protocol frame. Table 24 shows valid combinations of S bit and security-related TLVs.